National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

UNIVERSAL ISOCHRONOUS PAYLOAD MULTIPLEXING

BY

TIMOTHY M. FRIESEN ©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL, 1994

National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-95030-7

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Timothy M. Friesen**

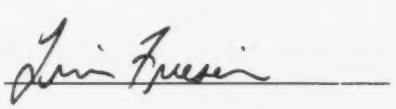
TITLE OF THESIS: **Universal Isochronous Payload Multiplexing**

DEGREE: **Master of Science**

YEAR THIS THESIS GRANTED: **1994**

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Timothy M. Friesen

10916 - 35 A Avenue

Edmonton, Alberta, Canada

T6J 0A3

DATE: July 26, 1994

To achieve all that is possible,

we must attempt the impossible--

To be as much as we can be,

we must dream of being more.

Karen Ravin

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **UNIVERSAL ISOCHRONOUS PAYLOAD MULTIPLEXING** submitted by **TIMOTHY M. FRIESEN** in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE.**

Dr. W.D. Grover (supervisor)

Dr. W.A. Krzymien

Dr. P.A. Goud

Dr. A. Basu

DATE: July 19, 1994

ABSTRACT

The public telecommunications network is a sophisticated and complex system that can transport various forms of information. However, it provides no means to transport digital isochronous signals that do not rigidly conform to its standards. The ability to so has the potential of increasing applications for telecommunications service providers.

This thesis addresses this issue by describing a novel technique called *Universal Isochronous Payload Multiplexing* (UIPM). UIPM is as method that allows synchronous transport, over the public network, of digital signals that cannot normally be transported by such a facility. UIPM accomplishes this by providing a mechanism to adapt arbitrary-rate digital isochronous signals into the payload portion of an industry-standard telecommunications carrier.

Three major components of UIPM are developed in this thesis: the UIPM generic format, the UIPM format design algorithm, and experimental hardware. The UIPM generic format is a variable length plesiochronous multiplexing format whose final length is determined by the UIPM format design algorithm. Predicted jitter performance is used by the algorithm to determine which one of many feasible UIPM format sizes is optimum for adapting a given input signal into a desired carrier. Actual jitter performance of optimum UIPM formats is measured with the experimental hardware that can be configured to emulate key time-base functions of any UIPM format.

This thesis provides predicted jitter performance for signal rates ranging from 1 Mb/s to 44 Mb/s in 100 kb/s increments, adapted via UIPM into a DS3 payload. Over 95% of these signal rates resulted in worst case predicted jitter of less than 0.5 UI p-p. Also, it is shown that more than 99% of all signal rates between the theoretical minimum and maximum rates, in 1 b/s increments, can be accommodated by the UIPM format design algorithm. A comparison between predicted and measured results for a set of test rates provides proof-of-concept and verifies the principles of UIPM. Future developments

of UIPM enabling the simultaneous transport of multiple arbitrary-rate digital isochronous signals over one or more standard telecommunications carriers are also considered.

ACKNOWLEDGMENTS

I would first like to express sincere gratitude to my supervisor Dr. Wayne D. Grover for introducing me to the field of telecommunications; for providing me with an exciting and worthwhile thesis topic and showing confidence in my ability to conduct the necessary research; for steering an employment opportunity in my direction, and finally, for guiding and assisting me throughout my graduate program. I would also like to thank the members of my examining committee, Dr. Witek A. Krzymien, Dr. Paul A. Goud, and Dr. Anup Basu for taking the time to read and comment on this thesis.

Special thanks is also given to Anthony Soong for many useful conversations, his thorough review of an earlier draft of this thesis, and for his sage financial advice on investments and how to become a "wealthy barber".

The following individuals and institutions also deserve recognition and a special word of thanks:

- Mr. David Clegg for his assistance in the lab and knowledgeable advice on circuit design.
- TR*Labs* and the University of Alberta for financial assistance.
- The staff and administration of TR*Labs* for providing me with facilities enviable to most graduate students.
- My parents-in-law, Ted and Rita Rempel, for meals, babysitting, and other support.
- LSI Logic of Canada Inc. for providing me with employment and allowing me to continue with my graduate studies while fulfilling my position at TR*Labs* as the LSI Logic Industrial Affiliate.

Finally, I would like to express much deserved appreciation to my loving wife Marnie and my two energetic daughters, Rebecca and Marika, for their patience and endurance while I labored towards this goal, and for keeping major yard work and renovations at bay until I was finished.

CONTENTS

1 INTRODUCTION 1

1.1 Background 1

1.2 Applications for UIPM 5

1.3 Thesis Outline 7

1.4 Summary 8

2 REVIEW OF TELECOMMUNICATIONS STANDARDS AND INTRODUCTION TO UIPM 9

2.1 Overview of Telecommunications Standards 9

2.2 Standard Signal Transport Methods 13

2.2.1 The Plesiochronous Digital Hierarchy 13

2.2.2 Synchronous Optical Network (SONET) 16

2.2.3 Asynchronous Transfer Mode (ATM) 18

2.3 Jitter in Plesiochronous Multiplexing 20

2.4 Stuff Ratio 24

2.5 The Transport of Arbitrary-Rate Isochronous Payloads over the Public Network 25

2.6 UIPM System Concept 28

2.7 Scope and Contribution 32

2.8 DS3 Payload Rate 33

2.9 Summary 34

3 UIPM GENERIC FORMAT DESCRIPTION 36

3.1 Basic Principles 36

3.2 Detailed Description 38

3.3 Size Constraints 40

3.4 Operating Dynamic Range 42

3.5 Framing Time for Two Dimensional Search 44

3.6 Summary 48
4 UIPM FORMAT DESIGN ALGORITHM 49
4.1 Basic Principle 49
4.2 Design Criteria 50
4.3 Software Implementation 52
4.3.1 General Description 53
4.3.2 Pseudocode of Format Design Algorithm 59
4.3.3 Flowchart of Format Design Algorithm 62
4.4 Waiting Time Jitter Modeling 62
4.4.1 General Description 62
4.4.2 Flow Chart of Jitter Simulation Algorithm 64
4.4.3 Desynchronizer PLL Model 66
4.4.4 Simulation Details 68
4.5 Validation of Jitter Modeling 72
4.6 Summary 74
5 EXPERIMENTAL HARDWARE 76
5.1 Clock gapper 78
5.1.1 Variable Stuffing Mechanism 80
5.1.2 Programmable UIPM Clock Gapping Logic 81
5.1.3 Verification of Clock Gapping Circuitry 83
5.2 UIPM Desynchronizer PLL 84
5.2.1 PLL Design 86
5.2.2 Practical Design Considerations 90
5.2.3 Hardware Modules 91
5.2.3.1 Analog Block 91
5.2.3.2 Divider Block 92
5.2.3.3 VCO Block 92

5.3 Hardware Performance 95
5.4 Summary 98
6 RESULTS 100
6.1 Predicted Performance 100
6.1.1 No-Format Rates 102
6.1.2 General Analysis and Observations 109
6.1.3 Selection of Experimental Test Rates 121
6.2 Measured Performance 127
6.3 Summary 139
7 FUTURE WORK 142
7.1 UIPM Generic Format Structure 142
7.1.1 Definition of the Subframe Overhead Bits 143
7.1.2 Other Generic Format Modifications 149
7.2 UIPM Format Design Algorithm 145
7.2.1 Efficiency, Format Size, and Jitter Performance 146
7.2.2 Other General Modifications to the Algorithm 147
7.2.3 Algorithm Modifications for n/m UIPM Systems 148
7.3 Complete UIPM System Hardware 150
7.3.1 Existing UIPM Experimental Hardware 150
7.3.2 Phase 1 UIPM Multiplexer 155
7.3.3 Phase 1 UIPM Demultiplexer 158
7.3.4 Hardware Considerations for Future UIPM Systems 162
7.4 Summary 165
8 CONCLUSIONS 166
8.1 Summary of Work 166
8.2 Final Conclusions 170
REFERENCES 171

APPENDIX A: Derivation of Stuff Ratio Equations 1-1 and 1-2 174
APPENDIX B: Source Code for the UIPM Format Design Algorithm 176
APPENDIX C: Derivation of Cutoff Frequency Expressions (Eqns. 4-8 and 4-9) 187
APPENDIX D: Second Order IIR LPF Design and Implementation 190
APPENDIX E: Schematics 194
APPENDIX F: Derivations for Minimum and Maximum Tributary Rates 216
APPENDIX G: Wideband p-p and rms Jitter 219
APPENDIX H: Glossary 225

LIST OF TABLES

Table

2.1 Comparison of attributes 27

2.2 Comparison of ATM and UIPM as service adapters 28

3.1 Fixed and variable dimensions of the UIPM generic format 41

4.1 Intermediate results of UIPM format design algorithm for 10 Mb/s case 57

4.2 Terms and definitions used in pseudocode 58

5.1 UIPM desynchronization versus conventional desynchronization 77

5.2 Measured versus theoretical results verifying correct operation of the clock gapper 84

6.1 No-format rates 105

6.2 Experimental design parameters 128

6.3 Comparison of predicted and measured results 129

LIST OF FIGURES

Figure

2.1 STM and ATM methods 11

2.2 Plesiochronous digital multiplexing hierarchy 14

2.3 The DS1 signal format 14

2.4 DS2 format 15

2.5 DS3 format 16

2.6 SONET STS-1 frame 17

2.7 ATM layers of the BISDN protocol stack 18

2.8 Generic ATM cell 19

2.9 Using SONET for the transport of ATM cells 20

2.10 Synchronizer employing pulse stuff justification 21

2.11 Example of pulse stuffing synchronization 22

2.12 Desynchronizer employing a Phase-Locked Loop (PLL) 23

2.13 ATM and UIPM depicted as interfaces for user signals 28

2.14 Implementation of UIPM 29

2.15 UIPM process 30

2.16 n/m UIPM system 31

2.17 UIPM block diagram 31

3.1 Generic Format structure 38

3.2 Framing and decoding process 40

3.3 DS3 frame structure 42

3.4 Variable bandwidth utilization by a UIPM signal 44

3.5 pdf of an unsuccessful maximum length search 46

Figure

3.6 Increase in framing time for two dimensional framing 47

4.1 Combined tributary-carrier frequency tolerance criterion 52

4.2 Predicted jitter performance of UIPM formats based on p-p jitter 55

4.3 Predicted jitter performance of UIPM formats based on rms jitter 55

4.4 UIPM format design algorithm flowchart 61

4.5 Flow chart for jitter simulation algorithm 65

4.6 Direct Form I realization of IIR digital filter 68

4.7 Frequency response of ideal desynchronizer PLL (equation 4-6) 71

4.8 Frequency response of laboratory desynchronizer PLL (equation 4-7) 71

4.10 Predicted jitter versus DS1 offset frequency using jitter model described in thesis 73

4.11 Predicted jitter versus DS1 offset frequency, from [MoKG91]: a) bit-by-bit sampling, b) frame-synchronous sampling 73

5.1 Experimental hardware block diagram 76

5.2 Clock gapping 78

5.3 Clock gapper block diagram 79

5.4 Phase comparison method used in variable stuffing mechanism 80

5.5 Simplified block diagram of UIPM clock gapping logic 81

5.6 Waveforms in desynchronizer PLL 85

5.7 Use of dual +N blocks for proper phase detection 86

5.8 PLL block diagram 87

5.9 Simplified description of the PLL a) Functional block diagram, b) Linear model of the PLL 87

5.10 Transfer function of the PD/LPF circuit 91

Figure

5.11 VCO block configurations
a) VCO block based on MC4024 VCM (1-10 MHz range)
b) VCO block based on MC1648 VCO (10-40 MHz range) 93

5.12 Transfer function of the MC4024 VCO for Cx=56pf and Cx=220pf 94

5.13 Transfer function of the MC1648 VCO for Cx=0 and Cx=220pf 94

5.14 Equipment setup 95

5.15 Performance waveforms (f_t=10 MHz)
a) Original (top) and recovered (bottom) tributary clock
b) Original (top) and gapped (middle) carrier payload clock 96

5.16 Performance waveforms (f_t=20 MHz)
a) Original (top) and gapped (middle) payload clock for ρ=0
b) Original (top) and gapped (middle) payload clock for ρ=1 97

6.1 Predicted UIPM jitter performance for tributaries mapped into a DS3 payload
a) using ideal PLL model with Butterworth type LPF characteristic
b) using model of PLL constructed for experimentation 101

6.2 UIPM jitter performance of tributaries near a no-format region 103

6.3 Tributary and format offset ranges for no-format rates
a) $f_t=f_p(x/y)$, b) $f_t \approx f_p(x/y)$ 108

6.4 Case where a large format can have less jitter than a small one
a) a small format can have high jitter if the tributary offset boundary is too close to a ρ=0 condition
b) a larger format will have less jitter because the tributary offset boundary is farther away from a ρ=0 condition 111

6.5 Maximum predicted jitter versus format size for 1 Mb/s case 113

6.6 Maximum predicted jitter versus format size for 5 Mb/s case 113

6.7 Maximum predicted jitter versus format size for 5.5 Mb/s case 114

6.8 Maximum predicted jitter versus format size for 6.5 Mb/s case 114

Figure

6.9 Maximum predicted jitter versus format size for 8.8 Mb/s case 115

6.10 Maximum predicted jitter versus format size for 10 Mb/s case 115

6.11 Maximum predicted jitter versus format size for 11.1 Mb/s case 116

6.12 Maximum predicted jitter versus format size for 20 Mb/s case 117

6.13 Maximum predicted jitter versus format size for 25.26 Mb/s case 117

6.14 Maximum predicted jitter versus format size for 29.5 Mb/s case 118

6.15 Maximum predicted jitter versus format size for 30 Mb/s case 119

6.16 Maximum predicted jitter versus format size for 36.2 Mb/s case 119

6.17 Maximum predicted jitter versus format size for 37.9 Mb/s case 120

6.18 Predicted jitter for a 1 Mb/s tributary over a +/-100 ppm offset range (level 3 UIPM format) 122

6.19 Predicted jitter for a 5 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format) 122

6.20 Predicted jitter for a 10 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format) 123

6.21 Predicted jitter for a 20 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format) 123

6.22 Predicted jitter for a 30 Mb/s tributary over a +/-100 ppm offset range (level 3 UIPM format) 124

6.23 Predicted jitter for a 36.2 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format) 124

6.24 Predicted jitter for a 25.26 Mb/s tributary over a +/-100 ppm offset range 125

6.25 Predicted jitter for the M12 case over a +/-130 ppm offset range 126

6.26 Graphical comparison of predicted and measured results 130

6.27 Comparison of predicted and measured jitter for a 10 Mb/s case 131

6.28 Comparison of predicted and measured jitter for a 25.26 Mb/s case 132

Figure

6.29 Original (top) versus recovered (bottom) tributary clock for 1 MHz case 134

6.30 Original (top) versus recovered (bottom) tributary clock for 5 MHz case 134

6.31 Waveforms for 10 MHz case
a) Original (top) versus recovered (bottom) tributary clock
b) Top: original tributary clock. Bottom: output of PLL with original tributary clock as input, showing intrinsic jitter 135

6.32 Original (top) versus recovered (bot) tributary clock for 20 MHz case..... 136

6.33 Waveforms for 25.26 MHz case
a) Original (top) versus recovered (bottom) tributary clock, with N=9
b) Original (top) versus recovered (bottom) tributary clock, with N=31..... 137

6.34 Original (top) versus recovered (bot) tributary clock for 30 MHz case..... 138

6.35 Original (top) versus recovered (bot) tributary clock for 36.2 MHz case..... 138

7.1 An example of a multi-frame UIPM generic format 149

7.2 VCO block based on down-conversion method (1-50 MHz range) 151

7.3 Transfer function of the wide ratio VCO 151

7.4 Analog block with N-compensator 153

7.5 Alternate PLL design 154

7.6 Phase 1 prototype of UIPM multiplexer 156

7.7 Adding carrier overhead to a UIPM signal
a) method 1, b) method 2 157

7.8 Demultiplexing a UIPM signal from a standard carrier
a) method 1, b) method 2 159

7.9 Phase 1 prototype of UIPM demultiplexer..... 160

7.10 Framing and clock gapping logic in a UIPM demultiplexer 161

7.11 Multiplexer portion of an n/m UIPM system where n=m=3
a) Three 3/1 UIPM multiplexers with selectable inputs
b) Detail of a 3/1 UIPM multiplexer 164

Figure

8.1 UIPM generic format structure 167

G1 Phase waveform for 10 Mb/s to DS3 payload mapping
a) Plot of first 500 frames
b) Zoom in view 220

G2 Phase waveform for M12 case
a) Plot of first 500 frames
b) Zoom in view 221

G3 Sawtooth waveform 224

LIST OF SYMBOLS

Δ	combined tributary and synchronous carrier frequency tolerance in ppm
Δ_p	frequency tolerance in ppm of the carrier payload rate
Δ_s	frequency tolerance in ppm of the synchronous carrier
Δ_t	frequency tolerance in ppm of the tributary signal
$\Delta\phi_D$	relative change in phase between the tributary and carrier payload clocks during the carrier payload clock period when a time slot contains a data bit
$\Delta\phi_O$	relative change in phase between the tributary and carrier payload clocks during the carrier payload clock period when a time slot contains an overhead or stuff bit
ϕ_i	*i*th end-of-frame phase sample representing the difference in phase at that time between the tributary and carrier payload clocks
$\Phi_{in}(s)$	Laplace transform of the input phase to a PLL system
$\Phi_{out}(s)$	Laplace transform of the output phase to a PLL system
ρ	stuff ratio
σ	standard deviation
τ	time constant
τ_1, τ_2	time constants of the LPF in a PLL
ω	angular frequency with units of radians per second
ω'	pre-warped angular frequency
ω_c	angular cutoff frequency at which a system's magnitude response is 3 dB lower (half power point) than at 0 frequency
ω_n	angular natural frequency of a second order system
ζ	damping coefficient

a_0, a_1, a_2	coefficients in the numerator of the transfer function for an IIR discrete-time filter
b_0, b_1, b_2	coefficients in the denominator of the transfer function for an IIR discrete-time filter
d	total number of data bits in one frame of a UIPM format
d_{min}	minimum number of data bits in one frame of a UIPM format
elmax	maximum number of bits allowed in a data element in the UIPM generic format
eltot	total number of bits currently existing in a data element
F(s)	transfer function of a LPF in the s-domain
f_b	bit rate
f_c	cutoff frequency (in Hz) at which a system's magnitude response is 3 dB lower (half power point) than at 0 Hz
f_{DS2}	DS2 bit rate (6.312 Mb/s)
f_{DS3}	DS3 bit rate (44.736 Mb/s)
f_f	frame rate
f_g	gapped clock signal
fixed	number of fixed stuff bits per frame of a UIPM format
f_n	natural frequency (in Hz) of a second order system
f_p	synchronous carrier payload rate (b/s)
f_r	read clock rate (Hz)
f_s	synchronous carrier rate (b/s)
$f_s max$	maximum allowed carrier rate based on its frequency tolerance
$f_s min$	minimum allowed carrier rate based on its frequency tolerance
f_{so}	variable stuff opportunity rate (Hz)
f_t	tributary rate (b/s)

$f_t max$	maximum allowed tributary rate based on its frequency tolerance
$f_t min$	minimum allowed tributary rate based on its frequency tolerance
f_{vco}	frequency (in Hz) of the VCO output
f_w	write clock rate (Hz)
H(s)	transfer function of a PLL system in s-domain
H(z)	transfer function of a discrete-time filter in z-domain
k	number of fixed stuff (R) bits in a data element
K_ϕ	conversion gain of a phase detector
K_v	conversion gain of a VCO
ℓ	current framing interval length (in bits) for which frame alignment is being sought
L	true framing interval length (in bits) for which frame alignment is possible
L_{max}	maximum framing interval length (in bits)
L_{min}	minimum framing interval length (in bits)
N	integer division value
n	number of data (D) bits in a data element
numel	number of elements in one frame of a UIPM format
oh	number of dedicated overhead bits in a multiplexing format
so	number of variable stuff opportunities in the master frame of a format
submax	maximum size in bits of a UIPM generic format's subframe
T	total number of bits per frame in a UIPM format
$t_{ave\ max}$	average maximum length search time (in seconds) for conventional framing

$t_{ave\ max\ 2D}$	total average maximum length search time (in seconds) for a two dimensional search procedure
thresh	phase threshold (UI) between the tributary and carrier payload clocks that initiates a stuff request
t_{inc}	increase in time two dimensional framing will take over conventional framing
T_{max}	maximum frame size (in bits) of the UIPM generic format
t_r	read clock period
T_s	sampling period
t_w	write clock period
warp	pre-warping expression
x_n	input sample to a discrete-time filter
y_n	output sample from a discrete-time filter

LIST OF ABBREVIATIONS AND ACRONYMS

AAL	ATM Adaptation Layer
ADC	analog to digital conversion
ATM	Asynchronous Transfer Mode
b/s	bits per second
BER	bit error rate
BISDN	Broadband Integrated Services Digital Network
BW	bandwidth
C1, C2, C3	control bits
Clk	clock
CPU	Central Processor Unit
CS	Convergence Sublayer within the AAL
D	data bit
DAC	digital to analog conversion
Dig. Atten.	digital attenuator
dB	decibel
DE	data extra count
DP	data pattern count
DSn	Digital Signal n, where n = 1, 2, 3, or 4
E	error correction bit
EMI	electromagnetic interference
F	framing bit
FDM	frequency division multiplexing
FEC	forward error correction

HEC	header error control field in an ATM cell
Hz	Hertz (one cycle per second)
i/p	input
IIR	Infinite Impulse Response
ISDN	Integrated Services Digital Network
kHz	kilohertz (1000 Hz)
LAN	local area network
LO	local oscillator
LPF	low pass filter
M	master framing bit
Mb/s	megabits (one million bits) per second
MHz	megahertz (one million Hz)
ms	milliseconds (10^{-6} seconds)
NB	narrowband
NISDN	Narrowband Integrated Services Digital Network
ns	nanoseconds (10^{-9} seconds)
o/p	output
OC-N	level N optical carrier
Od, Or, Om	UIPM overhead bits
OH	overhead (e.g.: general dedicated overhead bits)
P	pattern count
p-p	peak-to-peak
PCM	pulse code modulation
PD	phase detector

pdf	probability density function
PLL	phase-locked loop
PMD	Physical Medium Dependent sublayer (part of ATM physical layer)
POH	path overhead
ppm	parts-per-million
R	fixed stuff bit
rad/s/v	radians per second per volt
RE	extra fixed stuff bit count
RF	radio frequency
rms	root-means-square
RP	fixed stuff bit pattern count
s	second
S	variable stuff bit
SAR	Segmentation And Reassembly sublayer within AAL
SFET	Synchronous Frequency Encoding Technique
SFT	Search Fail Time
SO	(variable) stuff opportunity
SONET	Synchronous Optical Network
SPE	Synchronous Payload Envelope
STM	Synchronous Transfer Mode
STS-N	level N Synchronous Transport Signal
TC	Transmission Convergence sublayer (part of ATM physical layer)
TDM	time domain multiplexing
TOH	transport overhead

TR*Labs*	Telecommunication Research Laboratories
TTL	transistor-transistor logic
UI	unit interval
UIPM	Universal Isochronous Payload Multiplexing
VCI	virtual channel identifier label in an ATM cell
VCM	voltage controlled multivibrator
VCO	voltage controlled oscillator
VLSI	very large scale integrated circuit
VPI	virtual path identifier label in an ATM cell
v	volts
Ve	error voltage
v/rad	volts per radian
VT	Virtual Tributary
WB	wideband
WJT	waiting time jitter
Xtal Osc.	crystal oscillator

1 INTRODUCTION

1.1 BACKGROUND

Telecommunication is a dynamic, rapidly developing field. Advances over the past forty years have resulted in rapid and efficient transmission of voice and data over long distances. The development of digital technology and fiber optic transmission have been primarily responsible for these advances [BLab82, Mino91]. However, even as new standards such as the Synchronous Optical Network (SONET) and the Asynchronous Transfer Mode (ATM) are developed, advances in other fields of information technology suggest the need for even more flexible and efficient transmission systems.

Current technology allows transport of signals conforming to standard formats of the telecommunication industry. Other industries (such as television and computer industries), however, have developed signal formats which do not conform to these standards. Users and designers of equipment generating such signals may, therefore, benefit from a service which would allow them to transport nonstandard digital signal formats across the public telecommunication network in real time while maintaining synchronization at the application rate, with a minimum of signal delay and jitter.

Today's public network transports mainly voice and data. Virtually all signals in the public telecommunications network are digital *isochronous* signals conforming to a specified standard rate and format, and have a *plesiochronous* relationship to one another. Two (or more) signals have a plesiochronous relationship when their rates are nominally the same but may vary within specified limits (i.e. the frequency tolerance of the signal's rate). To describe digital isochronous signals, and their significance in this thesis, a distinction is made between isochronous signals and signals that are *asynchronous* in nature (typical of many data communication-based signals).

Isochronous digital signals are continuous-time (constant bit rate) signals whose significant instances (rising or falling edges) are multiples or sub-multiples of a local

reference clock. In other words, isochronous signals contain inherent timing information that occurs at a periodic rate. This timing information can be extracted at the destination and used to clock in data bits. Digital voice and video (uncompressed) are examples of isochronous signals. In general, isochronous signals are time and delay sensitive since corruption of timing destroys or alters the information and too much delay, as in voice over the telephone, becomes intolerable.

Asynchronous communications, in contrast, use signals that are not continuous-time signals and do not have a constant bit rate. Rather, they tend to exist as bursts of information consisting of packets or words containing a number of bits. These bursts of information are sent only when needed, down a channel common to other devices. Many LAN signals are asynchronous in nature. The asynchronous nature of these signals refers to the totally nonsynchronous and aperiodic relationship between the bursts of information, while within a word or packet the data bits occur at a regular (periodic) rate. For the duration of the packet the signal could be considered isochronous, but the packet rate is random.

As a result, signals in asynchronous communication do not carry inherent timing information on a continuous basis. Timing is achieved by the use of start and stop bits (as in the RS232 protocol) or a preamble of timing bits at the start of a packet. All devices exchanging information must agree upon a common data rate. Each device may have a local clock running at the same nominal rate, or use clock recovery techniques to regenerate a clock signal during the preamble and use this clock to read in the data bits of a packet. In the RS232 protocol, for example, the start bits tell the device when to start clocking in bits. Here the receiving device's clock does not have to be synchronous to the sending device's clock, or even the exact same frequency, as long as it does not erroneously clock bits for the duration of the word.

Asynchronous transmission is very efficient for typical data communication where file transfer is desired because each device has access to the full channel capacity for a

brief period of time and no one device can monopolize the channel. However, this method of transmission can result in other less desirable characteristics such as packet loss and long packet delay when the system is congested.

Asynchronous (packet based) information can be transported in an isochronous fashion by buffering the information and then transmitting it at a constant rate that is high enough to avoid buffer overflow. Framing bits (or some method of framing) need to be included so that the location of the information can be determined. Also, dummy bits or dummy packets need to be sent when no information is present in the buffer. The packets can then easily be received at the destination.

Asynchronous transmission of isochronous signals, however, is not easily accomplished. The random delay and arrival of packets, and the chance of lost packets, associated with asynchronous transmission, makes it very difficult to transport isochronous signals in an asynchronous fashion. As a result, the periodicity of the isochronous signal is destroyed and timing information is lost.

The transport of isochronous signals over the public network is presently accomplished by using time division multiplexing (TDM) techniques that fall under a more general description known as Synchronous Transfer Mode (STM) (details of STM are discussed in Chapter 2). The main disadvantage of STM is that only signals conforming to a specified standard rate and format can be transported. Therefore the transportation of *arbitrary-rate* digital isochronous signals over the public network is not possible. The term *arbitrary-rate* refers to signals that may have any particular fixed rate (within a specified tolerance) and may bear no relationship in rate or format to the standard signals expected by the public telecommunications network. The usual solution to this problem is to design dedicated equipment to convert a particular nonstandard signal into a standard signal acceptable to the network. But dedicated equipment for every different nonstandard signal desiring transport is costly.

Another method of transporting signals over the public network currently under development is called the Asynchronous Transfer Mode (ATM). This is a high speed cell-based (fixed packet length) switching and multiplexing technique for transporting data communications signals in an asynchronous manner [Mino91, Lebo92]. ATM makes more efficient use of channel capacity than does STM, and can accommodate signals of various rates, but ATM suffers from the potential problems cited earlier of transporting digital isochronous signals such as voice and video.

The inability of STM and ATM to adequately transport arbitrary-rate digital isochronous signals is further demonstrated by the following example. A serial isochronous digital signal is generated as a computer backplane source at 40 Mb/s. A customer wishes to send this signal to a distant computer without a reduction in rate, and without the cost of signal delay and protocol overhead associated with packet transfer methods which treat the signal as a large data file. In fact, this signal is needed as a bit timing reference for the far end equipment and therefore cannot afford to lose the inherent clocking information. Surely the public network could move this 40 Mb/s signal using its fiber optic transport systems. From the previous discussion it is clear that this signal *cannot* be transported in its native form as a 40 Mb/s serial data stream via the present public network. In fact, today, and for the indefinite future, the only isochronous signal rates compatible with the public network are specifically 1.544, 6.312, 44.736, 51.84 Mb/s, and specific multiples of 51.84, Mb/s. How then, could someone with an arbitrary-rate isochronous digital signal interconnect pieces of equipment, of their own proprietary rates, via the public network?

An original technique to address this problem is developed in this thesis and is based on a concept proposed in [Grov89]. This technique, called *Universal Isochronous Payload Multiplexing* (UIPM), employs an algorithm to design a near optimum plesiochronous multiplex format for arbitrary-rate isochronous digital signals. The algorithm is used to configure programmable multiplex hardware to multiplex a customer's

signal with UIPM system bits to form the payload portion of a standard public network carrier signal. The *universality* of UIPM lies in its ability to accommodate isochronous digital signals of virtually any rate. This thesis presents a proof-of-concept of the UIPM system and provides original analyses of its performance. In addition, a number of contributions are made toward the practical realization of UIPM.

1.2 APPLICATIONS FOR UIPM

Applications for UIPM range from real time mainframe computer interconnection to a video interface for the television and film industry. Reduced costs for transmission of nonstandard payloads and flexible equipment could provide new markets for telecommunication service providers in instances where certain payloads are not transportable over today's network, or may not be well served by future ATM transport.

An illustrative example of the need for a UIPM-type service is the case of a super computer vendor wished to link its 88 Mb/s backplane level bus between its computers at facilities in Edmonton and Calgary via the public network. They discovered that this could not be done because their signal did not conform to one of the standard signal formats of the today's network. Moreover, even had ATM been available it would not have met their requirements because of the problems associated with clock recovery, packet loss, etc., discussed earlier.

TR*Labs* took an interest in this particular problem and designed a network interface to accomplish their goal. The network interface used a specific new format design that adapted the 88 Mb/s isochronous payload to two DS3 signals [Grov88]. The concept of UIPM arose from this development as a generalization of the one-off design conducted to serve the 88 Mb/s payload. This historical example serves to clarify the function and usefulness of UIPM technology. The full development of UIPM would allow telephone companies and telecommunications service providers to accommodate many

similar and latent applications while the end user would not have to acquire dedicated interface equipment.

To generalize the above example, the interconnection of mainframe computers over short distances may occur via a high speed backplane-level serial digital signal defined by an application specific format. Long distance interconnection of these computers over the public network is possible today only via modems or a digital data communications interface. However, typical modems are low speed and transceive asynchronously resulting in loss of any required synchronization. A data communications interface typically allows transfer at only 56 kb/s, 64 kb/s or up to a DS1 rate at best, and assumes that such an interface exists for the particular computers. In addition, data communications protocols are often packet based. UIPM technology would allow the transport of the backplane level signal in its native format without loss of rate or synchronization. This would occur via a UIPM hardware and software system design that is common to a large number of individual application problems.

The hypothesis is that a significant number of customers could benefit from transparent isochronous transport of their own nonstandard payloads over the network, but that many of these applications are not presently known. Consider, however, that other industries involved with electronic equipment have developed their own unique signal standards for intra-equipment use, storage, or in the form of custom LANs (e.g.: digital video, CD-ROM, MIDI, Ethernet, FDDI, computer bus standards, etc.).

In many cases, these standards evolved from historical and technological developments and do not correspond to telecommunication standards. For example, digitized video signals used in the television industry usually occur at rates that are multiples of the color subcarrier frequency (3.579545 MHz) - a rate entrenched in the historical development of television [BeBl86]. Current technology does not allow these customers to be served and full deployment of ATM transport services may not occur for a number of years. UIPM would provide a means of offering this service, without

requiring custom designed interface equipment or lengthy standardization procedures. This would open a new market for customers who have the applications to utilize the bandwidth available on the telecommunication network but cannot presently access it.

Other potential applications of UIPM technology include:

- transport of arbitrary-rate digital video for television broadcast facilities or video conference services
- real time image and video exchange between hospitals throughout the country for consultation and emergencies
- real time synchronous link from field to studio for remote recording and filming applications
- physical-layer interconnection of local area networks
- direct CEPT 2/3/4 (European digital hierarchy) interfaces to DS3 or SONET systems

1.3 THESIS OUTLINE

Chapter 2 provides background on telecommunications standards and details of standard signal transport methods, followed by a description of the UIPM system. Characteristics of the generic format used in the UIPM system are detailed in Chapter 3. Chapter 4 describes the rate adaptive format design algorithm and the method used for predicting jitter performance of UIPM formats. A description of the experimental hardware developed is given in Chapter 5. Results and analyses of computer simulations and experimental measurements of jitter performance are presented in Chapter 6. A comparison between predicted and measured jitter performance provide a proof-of-concept for the UIPM principle. Chapter 7 provides a discussion on further work necessary to make UIPM a reality and suggests changes to the UIPM system that may increase its effectiveness. Finally, Chapter 8 presents conclusions drawn from the results of the research.

1.4 SUMMARY

It was shown in this chapter that current (STM) and planned (ATM) telecommunication transport standards will not provide for the transmission of arbitrary-rate isochronous digital payloads. Such payloads are generated by industries which have developed their own signal standards or by individual designers with proprietary applications such as a computer manufacturer with an 88 Mb/s backplane level signal. These applications cannot utilize the available bandwidth on the public telecommunications network to transport their signals without expensive dedicated custom interface designs to adapt their signal to a standard rate and format. In the case of ATM, inherent loss of synchronization at the applications rate would also necessitate complex and possibly expensive equipment for clock recovery.

Universal Isochronous Payload Multiplexing (UIPM) is proposed as an alternative means through which a service can be offered to customers requiring real time rate-insensitive isochronous transport over long distances.

2 REVIEW OF TELECOMMUNICATIONS STANDARDS AND INTRODUCTION TO UIPM

This chapter presents a review of telecommunications standards and details of existing and planned transport methods such as the DSn plesiochronous digital hierarchy, SONET, and ATM. Jitter in plesiochronous multiplexing and the concept of stuff ratio are also reviewed. The issue of incompatibility between arbitrary-rate digital isochronous signals and telecommunications standards is addressed, followed by a description of the UIPM system concept. Subsequent chapters discuss details of UIPM system components. A discussion is then given about the UIPM components developed for the thesis project, which embodies the majority of the work presented in this thesis. Finally, a derivation is given for the DS3 carrier payload bit rate used for obtaining predicted and measured jitter values.

2.1 OVERVIEW OF TELECOMMUNICATIONS STANDARDS

Signal standards in the telecommunication industry have evolved over the last thirty to forty years. The industry has progressed from analog to digital systems and from copper to optical fiber as the transmission medium. This evolution has largely been due to technological advancements, an increase in the number of subscribers, and changes in subscriber demands [BLab82]. This section provides a general overview of telecommunications standards before specific details on standards relevant to this thesis are discussed in Section 2.2.

Initially, a Frequency Division Multiplexing (FDM) Hierarchy carried analog information consisting primarily of voice channels. This method has all but vanished, being superseded by a digital network. Advantages of a digital network over analog include reduced cost for purchase and maintenance of equipment, higher immunity to noise, general robustness, and the ability to carry data more effectively [BLab82, Mino91].

The current digital network operates in the Synchronous Transfer Mode (STM) using Time Division Multiplexing (TDM) [BLab82, Mino91, Stal92]. The signal between a user's telephone set and the local central office is currently analog. However, the rest of the inter-office network is almost completely digital. All transport between central offices conforms to a rigid set of standard digital signal formats. These formats accommodate voice traffic as well as certain restricted types of data signals.

The transition to a completely digital network will be realized with the Integrated Services Digital Network (ISDN). ISDN will result in enhanced telephone and data services to business and residential consumers. ISDN development consists of two phases: narrowband ISDN (NISDN) and broadband ISDN (BISDN). NISDN can accommodate payload rates that are specific multiples of 64 kb/s up to a maximum of the DS1 carrier rate (1.544 Mb/s). NISDN standards are currently well defined and operate with STM interoffice transport [Mino91, Stal92]. BISDN, however, includes a new basic mode of interoffice transport which is the Asynchronous Transfer Mode (ATM). Standards for BISDN include two rates, one at about 150 Mb/s, and one at about 600 Mb/s [Mino91, Minz89].

STM and ATM are the only two transport paradigms so far considered by the industry. STM is the basis of the current digital hierarchy standards, which use frame structures and plesiochronous multiplexing to multiplex and transport data over the network [Mino91, Minz89]. The digital signal standards (DS1, DS2, and DS3) were created to make efficient use of the limited bandwidth available in pre-1975 transmission media. This included twisted copper pairs (e.g., DS1 at 1.544 Mb/s and DS2 at 6.312 Mb/s), coaxial cable (e.g., DS3 at 44.736 Mb/s), electromagnetic waveguides, and the atmosphere in microwave transmission (e.g., DS3 rate) [Tele83]. Since the last decade, interoffice transport has occurred primarily over optical fiber carrying DS3 signals or multiple DS3 signals.

A new standard, called the Synchronous Optical Network (SONET), has recently been developed to accommodate the very high bandwidth capabilities of fiber optic communications [BaCh89, Ansi88a]. SONET was designed to create a completely synchronous infrastructure for the transport of telecommunication signals at greater than DS3 rates. SONET is a transport mechanism operating in STM but can carry ATM based signals by acting as the physical layer of the BISDN protocol. SONET also includes methods for adapting existing telecommunication standard formats (DS1-DS3) [Minz89]. The trend in the future will be to carry all telecommunications via fiber under the SONET standard. SONET equipment has already been developed and deployed in various locations across North America.

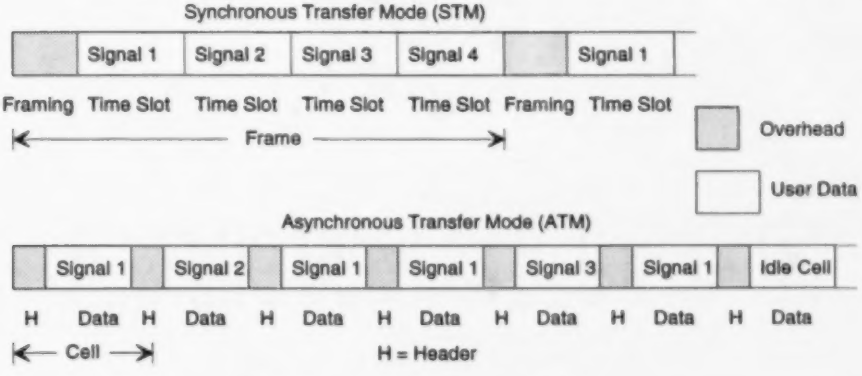


Figure 2.1 STM and ATM methods

Figure 2.1 shows the main difference between STM and ATM methods. STM uses time division multiplexing (TDM) to transport a number of synchronous (or plesiochronous) payload tributary signals over a common channel. Since the signals are either bit or byte interleaved, all signals must have a common synchronous bit rate for TDM to be effective. For plesiochronous tributaries, a method called *pulse stuffing justification* is used on each lower rate signal or tributary to raise it to a common synchronous rate before interleave multiplexing occurs. Pulse stuffing justification is

analogous to *space justification* utilized in word processors by adding appropriate space between words to achieve the desired margin alignment. (Section 2.3 provides more details on the mechanism of pulse stuffing justification and its effect on jitter.) In order to facilitate demultiplexing of these payloads, various framing and overhead bits are imbedded in the multiplex scheme. One important disadvantage of STM is its inability to multiplex payloads which do not conform to one of the standard rates and formats in the rigidly defined digital transmission hierarchy.

The Asynchronous Transfer Mode (ATM) is the newest member of public networking standards and is currently under development. ATM transports payload signals by converting payload data into packets or cells of information and multiplexing them on a statistical basis. Each cell contains the user data as well as overhead information contained in the header (H) for routing and error correction (see Figure 1.1). The statistical allocation of cells operates on a need basis with more cells generated for a higher rate signal than for a lower one [Mino91]. This allows ATM to accommodate signals of various rates, but results in loss of synchronization implicit in the original signal because the signal's periodicity is destroyed.

ATM operates well for applications where end-to-end synchronization is obtained by external means or where the application is of a file-transfer nature. However, loss of signal synchronization, the possibility of cell loss, and random cell delay may cause difficulties in using ATM to transport arbitrary-rate isochronous payloads. A method that addresses this problem is proposed in [Lau89]. The method described is called *Synchronous Frequency Encoding Technique* (SFET). The fundamental assumption of SFET is that both the source node and destination node are synchronized to the same clock. This assumption is satisfied only if SONET is used for the transport of ATM cells and network synchronization between SONET islands is guaranteed.

In the SFET method, the difference in rate between the source and the network is encoded and sent along with each cell. At the destination, the frequency difference

information is processed and the resulting difference frequency is subtracted from the network clock rate to obtain the original source rate. SFET also claims the ability to handle lost or misdelivered cells.

2.2 STANDARD SIGNAL TRANSPORT METHODS

2.2.1 The Plesiochronous Digital Hierarchy

In the existing digital hierarchy, plesiochronous digital signals are multiplexed together to form higher rate signals using TDM. Signals are combined on a hierarchical basis resulting in digital signal formats labeled as DSn, where n (n=0,1,2,3,4) indicates the level in the digital hierarchy. The formats describe how signals of a lower hierarchy are multiplexed together, along with necessary overhead control bits, to form a higher rate signal. Pulse stuff justification is used to bring each plesiochronous tributary to a slightly higher common synchronous rate after which multiplexing is accomplished by simple interleaving of the tributaries. Overhead and stuffing bits result in a DSn signal rate that is greater than the product of the number of tributaries and the nominal DS(n-1) tributary rate [Tele83, Ansi88b].

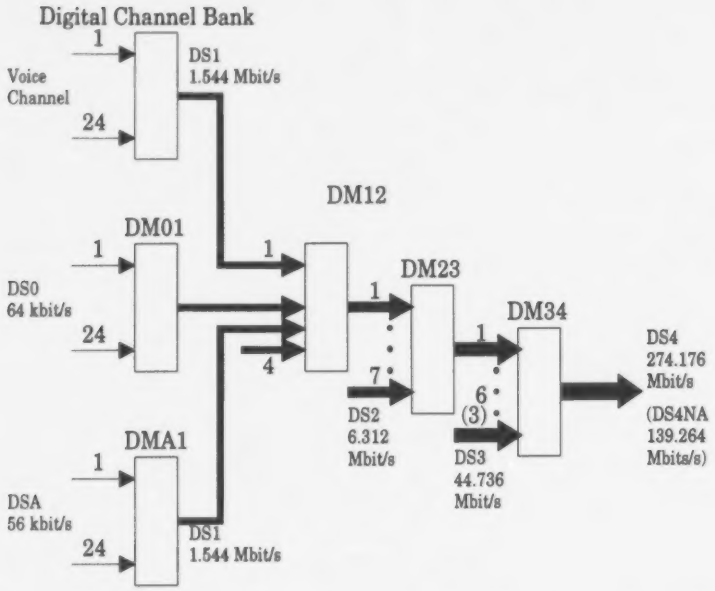


Figure 2.2 Plesiochronous digital multiplexing hierarchy [Tele83]

Figure 2.2 shows the plesiochronous digital multiplexing hierarchy. The lowest rate digital signal is the DS0 with a rate of 64 kb/s. A DS0 can consist of 64 kb/s data or, more commonly, a voice signal that has been converted from analog to digital form using 8 bit pulse code modulation (PCM) with a sampling rate of 8 kHz, resulting in 64 kb/s. The DS1 signal consists of 24 byte-multiplexed DS0 signals with one framing bit after every 24 sets of 8 tributary bits (192 bits). The DS1 frame occurs every 125 μs resulting in a rate of 1.544 Mb/s with a specified tolerance of ±130 parts-per-million (ppm). Figure 2.3 shows the DS1 signal format.

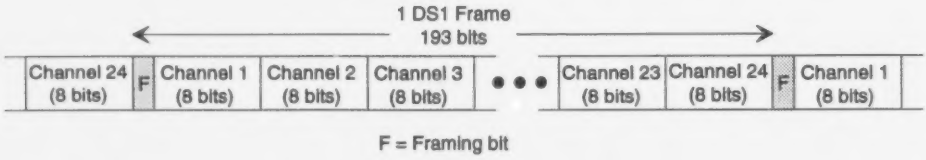


Figure 2.3 The DS1 signal format

The DS2 signal consists of 4 bit-multiplexed DS1 signals with one overhead bit (for framing or stuff indication) between every twelve sets of 4 tributary bits (48 bits). The DS2 signal rate is 6.312 Mb/s ±33 ppm. Figure 2.4 shows the DS2 format consisting of a master frame with 4 subframes. The master frame (M) bits form a framing pattern of 0111 to indicate start of master frame while the primary framing (F) bits form a repeating pattern of 01 used for frame alignment. The control (C) bits indicate the status of the variable stuff (S) bit for each frame. The S bit can carry either a data (D) bit or a dummy bit and is used in the pulse stuffing justification scheme to synchronize plesiochronous DS1 signals to a common synchronous rate before they are interleaved.

M_0 [48D] C_{11} [48D] F_0 [48D] C_{12} [48D] C_{13} [48D] F_1 [0D S_1 47D]
M_1 [48D] C_{11} [48D] F_0 [48D] C_{12} [48D] C_{13} [48D] F_1 [1D S_2 46D]
M_1 [48D] C_{11} [48D] F_0 [48D] C_{12} [48D] C_{13} [48D] F_1 [2D S_3 45D]
M_1 [48D] C_{11} [48D] F_0 [48D] C_{12} [48D] C_{13} [48D] F_1 [3D S_4 44D]

Figure 2.4 DS2 format

The DS3 signal consists of 7 bit-multiplexed DS2 signals with an overhead bit between every twelve sets of 7 tributary bits (84 bits). The rate for the DS3 signal is 44.736 Mb/s ±20 ppm. Figure 2.5 shows the DS3 format. The same general scheme is used as in the DS2 format except that here the primary framing pattern is 1001 and the master framing pattern is 010. There is also additional overhead bits: the X bits are used for alarm indication and the P bits are used to convey parity information.

X [84D] F_1 [84D] C_{11} [84D] F_0 [84D] C_{12} [84D] F_0 [84D] C_{13} [84D] F_1 [0D S_1 83D]
X [84D] F_1 [84D] C_{21} [84D] F_0 [84D] C_{22} [84D] F_0 [84D] C_{23} [84D] F_1 [1D S_2 82D]
P [84D] F_1 [84D] C_{31} [84D] F_0 [84D] C_{32} [84D] F_0 [84D] C_{33} [84D] F_1 [2D S_3 81D]
P [84D] F_1 [84D] C_{41} [84D] F_0 [84D] C_{42} [84D] F_0 [84D] C_{43} [84D] F_1 [3D S_4 80D]
M_0 [84D] F_1 [84D] C_{51} [84D] F_0 [84D] C_{52} [84D] F_0 [84D] C_{53} [84D] F_1 [4D S_5 79D]
M_1 [84D] F_1 [84D] C_{61} [84D] F_0 [84D] C_{62} [84D] F_0 [84D] C_{63} [84D] F_1 [5D S_6 78D]
M_0 [84D] F_1 [84D] C_{71} [84D] F_0 [84D] C_{72} [84D] F_0 [84D] C_{73} [84D] F_1 [6D S_7 77D]

Figure 2.5 DS3 format

The DS4 level exists at 274.176 Mb/s, but is not generally used in practice [BLab82, Tele83]. Another level 4 format standard exists and is denoted DS4NA where NA stands for North America. The DS4NA rate is 139.264 Mb/s and can multiplex 3 DS3 signals. This format has also not seen much use in the industry.

2.2.2 Synchronous Optical Network (SONET)

SONET is a standard defining a hierarchical family of optical transport signals with a synchronous frame structure and associated operations procedures [BaCh89, Ansi88a]. The SONET standard was developed jointly by the American and European standards bodies. American terminology is used in this discussion of SONET. The basic signal of SONET is the Synchronous Transport Signal level 1 (STS-1), which has a nominal rate of 51.840 Mb/s. Higher rate signals are achieved by byte-interleaving STS-1 signals into an STS-N frame structure where N=3,9,12,18,24,36,48,192. The STS-N always has a frame rate of 8 kHz [Mino91, Ansi88a].

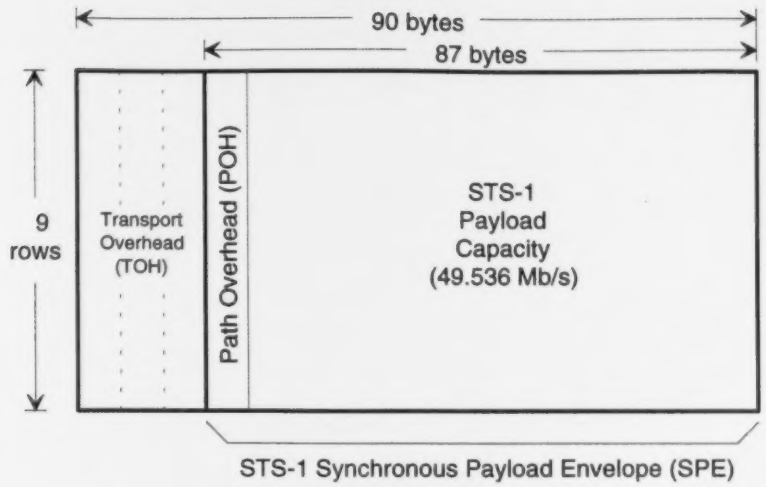


Figure 2.6 SONET STS-1 frame

Figure 2.6 shows an STS-1 frame structure consisting of transport overhead (TOH) and the Synchronous Payload Envelope (SPE). The TOH bytes are synchronous to the STS-1 frame while the SPE is not necessarily synchronous to an STS-1. Special payload pointers in the transport overhead monitor the start of an SPE and allow for transport of plesiochronous SPEs or for absorption of phase wander. The SPE consists of the payload capacity plus additional path overhead (POH) bytes that are synchronous to the SPE [Mino91, Ansi88a].

The STS-1 frame consists of 9 rows of 90 bytes. Thus a frame rate of 8 kHz results in an STS-1 rate of 51.84 Mb/s. The payload capacity of an STS-1 SPE is 49.536 Mb/s and can accommodate a DS3 signal (44.736 Mb/s), for which a special mapping format exists. For payload rates much less that the STS-1 SPE rate, a structure called a Virtual Tributary (VT) is used. To map sub-STS-1 SPE rate signals into VTs, pulse stuffing techniques similar to those used in the digital hierarchy are implemented [Ansi88a].

Because of the tremendous bandwidth available in optical fiber systems, the need to make SONET bandwidth efficient has not been necessary. This has allowed designers

of SONET to incorporate an overhead structure enabling easy access to individual synchronous payloads. Similar access to individual tributaries does not exist in the plesiochronous digital hierarchy due to pulse stuffing at each multiplex stage [BaCh89].

2.2.3 Asynchronous Transfer Mode (ATM)

The ATM standard was developed as part of the BISDN effort [Stal92]. ATM exists as the lower 3 layers in the BISDN protocol stack as shown in Figure 2.7.

<table>
<tr><td colspan="2">APPLICATION LAYER</td></tr>
<tr><td colspan="2">HIGHER LAYERS</td></tr>
<tr><td>Convergence Sublayer (CS)</td><td rowspan="2">ATM ADAPTATION LAYER (AAL)</td></tr>
<tr><td>Segmentation & Reassembly (SAR) Sublayer</td></tr>
<tr><td colspan="2">ATM LAYER</td></tr>
<tr><td>Transmission Convergence (TC) Sublayer</td><td rowspan="2">PHYSICAL LAYER</td></tr>
<tr><td>Physical Medium Dependent (PMD) Sublayer</td></tr>
</table>

Figure 2.7 ATM layers of the BISDN protocol stack

The ATM adaptation layer (AAL) consists of a convergence sublayer (CS) and a segmentation and reassembly (SAR) sublayer which together convert user data into 48 byte payload cells. The ATM layer adds a 5 byte header and produces an ATM cell for transfer to the physical layer. The physical layer consists of a transmission convergence (TC) sublayer and a physical medium dependent (PMD) layer. The TC sublayer generates and receives transmission frames (e.g. SONET) and is responsible for processing all overhead associated with the transmission frame (e.g. TOH and POH). The TC sublayer also packages cells inside the transmission frame and performs functions such as calculation of the header error control (HEC) information which is one byte of the 5 byte cell header. The PMD sublayer is the actual transport method, such as SONET, used to

transport ATM cells over the network.

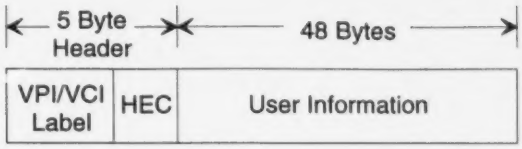


Figure 2.8 Generic ATM cell

Figure 2.8 shows the structure of an ATM cell. ATM cells are 53 bytes in length. The first 5 bytes are used for header information and the remaining 48 carry payload data. The *asynchronous* nature of ATM refers to the totally non synchronous relationship between cells carrying payload data from the same source, and between the rate of source transmission and the physical carrier signal. The 5 byte header consists mainly of the ATM address made up of the virtual path identifier (VPI) and the virtual channel identifier (VCI).

Two options have been considered for physical layer transmission [Stal92]. One option is to transmit a continuous stream of cells with no multiplex frame structure imposed at the interface. This option is termed *native ATM* in this thesis. In this case, the physical signal (but not the packetized payload data) is an isochronous signal. Synchronization is achieved by normal clock recovery techniques, and the establishment of cell boundaries is found by continuously monitoring the header error control (HEC) field in the cells.

The second option is to slot the cells into a SONET structure such as an STS-3. Currently this is the preferred option and methods for implementing it have already been developed. Figure 2.9 illustrates the process that occurs when SONET is used as the physical layer [Lane92].

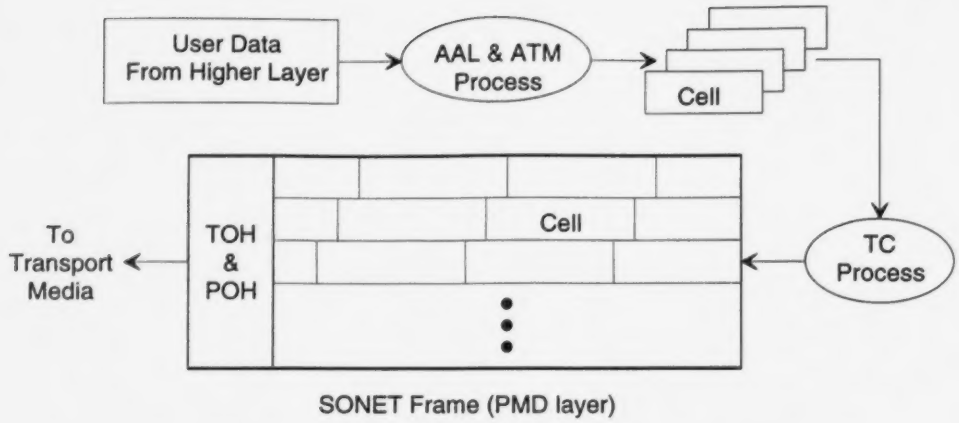


Figure 2.9 Using SONET for the transport of ATM cells

2.3 JITTER IN PLESIOCHRONOUS MULTIPLEXING

Jitter is the deviation of a signal's phase from its nominal value, resulting in symbol timing phase noise. For an isochronous digital signal, this results in the fluctuation in time of its transition edges compared to the edges of an ideal clock at the same nominal rate. This is equivalent to undesired pulse position or phase modulation of the digital signal. Jitter is measured in unit intervals (UI) where one UI equals the time period of one clock cycle. Alternatively, jitter is expressed as degrees of phase relative to 360^{o} for 1 UI. Excessive clock jitter results in bit errors because data bits are erroneously detected [BLab82]. Jitter performance is important in the design of plesiochronous multiplexing systems in general and also for the UIPM techniques discussed in this thesis.

The main cause of jitter in the multiplexing schemes previously discussed is pulse stuffing techniques, or pointer adjustments in SONET, used to multiplex plesiochronous signals [BLab82, TrVa89]. The following description of multiplexing and demultiplexing in the plesiochronous digital hierarchy explains the sources of jitter and techniques for jitter reduction, and serves as background for similar techniques used in UIPM.

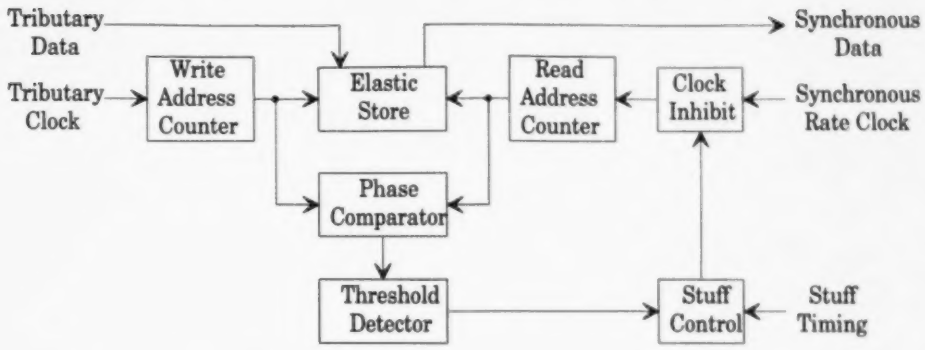


Figure 2.10 Synchronizer employing pulse stuff justification

Figure 2.10 shows how each plesiochronous tributary to be multiplexed is written into an elastic store or buffer. The write clock is derived from the plesiochronous signal itself using typical clock recovery circuits. The elastic store is read out at a synchronous rate that is faster than the highest possible plesiochronous tributary rate. To prevent the read operation from overlapping with the write operation, the read clock is inhibited at a predetermined time slot, called a stuff opportunity, whenever the phase difference between the read and write clocks crosses a specified threshold. To maintain the higher synchronous rate during this time, a dummy or *stuff* bit is inserted in place of the data bit not read. The contents of the stuff opportunity is conveyed by control bits in the format so that at the demultiplexer the bit in the stuff opportunity is not read if it is not a data bit. Stuff bits cannot be inserted immediately after the phase threshold is exceeded, otherwise their location would be uncertain to the demultiplexer. Rather, a stuff request is initiated, and a stuff bit inserted in place of data at the next available stuff opportunity time slot. Plesiochronous tributary signals are thus increased to a higher synchronous rate where they are interleaved together along with overhead bits peculiar to the format before being transported across the network [BLab82, Dutt72]. Figure 2.11 shows conceptualized bit streams before and after synchronization.

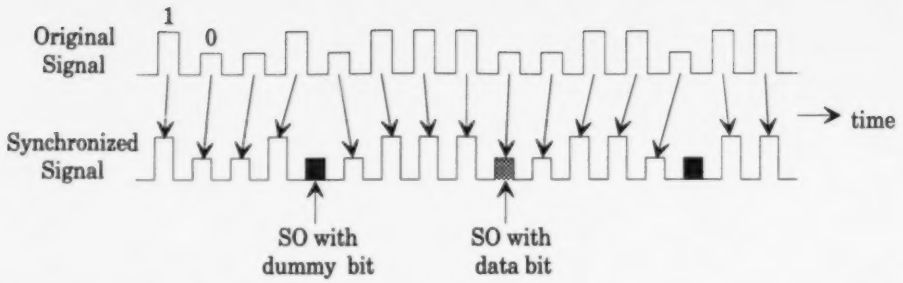


Figure 2.11 Example of pulse stuffing synchronization (SO = stuff opportunity)

At the demultiplexer, the reverse operation separates the tributaries and retrieves the data. The carrier signal is first deinterleaved into the original synchronized tributaries. The data of each tributary is written into an elastic store at a higher instantaneous rate than it is read out. The write clock is at the higher synchronous rate that existed prior to multiplexing of the tributaries, but is inhibited or gapped during stuff opportunities that carried a stuff bit. Thus, stuff bits are omitted and only payload data is written into the receiver elastic store. The read clock is derived by passing the gapped write clock through a phase-locked loop (PLL) which smoothes out the gaps and creates a clock whose rate is the average of the gapped clock rate. In other words, the read clock rate is the difference between the ungapped write clock rate (higher synchronous rate) and the average rate of gapping [BLab82, Dutt72, TrVa89]. This process occurs in the *desynchronizer* portion of the demultiplexer and is depicted in Figure 2.12.

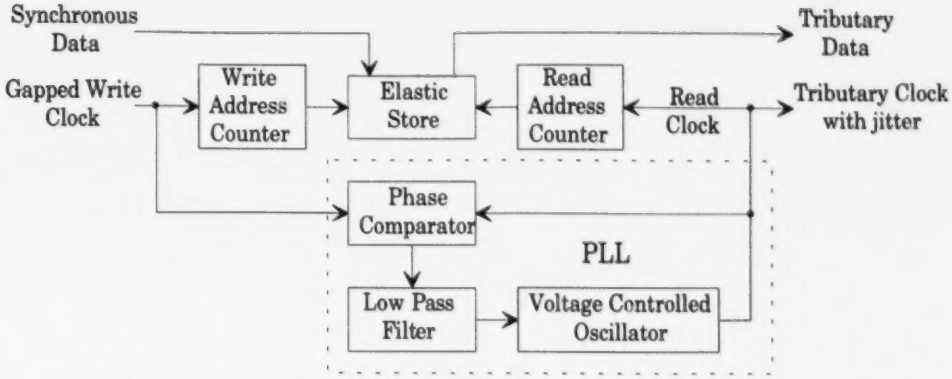


Figure 2.12 Desynchronizer employing a Phase-Locked Loop (PLL)

The generation of jitter is mainly due to pulse stuffing operations. This results from the random waiting period that exists between a stuff request and the next available stuff opportunity. This is known as *waiting time jitter* (WTJ) and is the predominant jitter resulting from multiplexing.

Waiting time jitter results in timing errors associated with the recovered read clock. Duttweiler [Dutt72] and Chow [Chow73] have analyzed waiting time jitter in terms of its spectral components and found that jitter contains random low frequency components. Some of this jitter is filtered out by the demultiplexer PLL since it acts as a low pass filter for phase noise; however, it is impossible to eliminate all of it. The amount of low frequency jitter noise power that is allowed to be generated by a plesiochronous multiplex system is specified in the standard of the format [Ansi87, Bell86]. As an example, the jitter specification for signals based on the DS3 format states that tributary jitter shall not exceed 1 UI p-p or 0.3 UI rms for a desynchronizer bandwidth of 350 Hz [Bell86].

2.4 STUFF RATIO

An important variable of the synchronization process is the stuff ratio (ρ), defined as the ratio of stuff opportunities stuffed with dummy bits to the total number of stuff opportunities available [BLab82]. A unique stuff ratio exists for any given frequency offset of a tributary mapped into a synchronous carrier.

The stuff ratio has been used to predict jitter performance of mapping formats and indicates the ability of a mapping format to accommodate signal tolerances [BLab82]. Duttweiler [Dutt72] and Matsuura et al [MaKY68] show that jitter is lowest when the stuff ratio is an irrational fraction and that peak to peak jitter is approximately 1/q UI when the stuff ratio is p/q. However, these analyses ignore the effect of overhead bits in the mapping format and assume each stuff opportunity has an equal probability of being stuffed. Although the stuff ratio may theoretically predict adequate jitter performance of a mapping format, results described in [GrMo86] and [GrMM87] show that the actual jitter performance at nominal rates may be significantly different. The effect of added overhead does not increase the amount of jitter but instead skews the relationship between jitter and the frequency offset of the tributary due to its tolerance. This justifies the use of simulation to predict jitter performance in cases where there is significant fixed overhead in the multiplex format.

The stuff ratio for any plesiochronous positive pulse stuffing mapping format can be calculated using the following expression:

$$\rho = \frac{d + so - \frac{N * f_t * T}{f_s}}{so}, \quad 0 \le \rho \le 1 \qquad (2\text{-}1)$$

where d is the number of time slots in a master frame dedicated for carrying tributary data bits; so is the number of stuff opportunities per master frame; T is the total number of bits per frame; N is the number of tributaries per frame; f_t is the tributary input rate; and f_s is the synchronous output carrier rate. Equation 2-1 is equivalent to expressions in

[BLab82] and other literature. Appendix A shows the derivation of Equation 2-1 from the principle definition of stuff ratio.

The stuff ratio can only range from 0 to 1. When $\rho = 0$, f_s-f_t is at its minimum that can be handled by the mapping format and, in this case, all stuff opportunities are used for information bits. When $\rho = 1$, f_s-f_t is at its maximum and all stuff opportunities are stuffed with dummy bits. As an example, one master frame of the DS3 format contains 4697 fixed data bits and 7 variable stuff opportunities with a total of 4760 bits, and carries 7 DS2 tributaries, resulting in a nominal stuff ratio of 0.3906.

When the number of stuff opportunities in a format is one per (master) frame (implying one tributary), the expression for the stuff ratio in Equation 2-1 simplifies to:

$$\rho = 1 - \mathrm{frac}\left(\frac{f_t}{f_f}\right) \tag{2-2}$$

where frac(x) is the fractional part of x and f_f is the frame rate of the format. This is the form used in this thesis for describing the stuff ratio of local formats generated by the UIPM algorithm. The derivation of Equation 2-2 is also presented in Appendix A. Using the DS3 case again as an example, the stuff ratio can found from Equation 2-2 by considering only one of the 7 frames in the DS3 master frame (and pretending it carries only one DS2 rather than 7 multiplexed DS2s). In this case the tributary rate remains the same (6.312 Mb/s for one DS2) and the frame rate is same as the DS3 master frame rate of 9398.32 Hz. The result is again a stuff ratio of 0.3906.

2.5 THE TRANSPORT OF ARBITRARY-RATE ISOCHRONOUS PAYLOADS OVER THE PUBLIC NETWORK

Standards are necessary in any industry to ensure compatibility among equipment manufacturers in achieving some common objective. The signal rates and formats acceptable to the public telecommunications network must conform to digital hierarchy

standards or SONET standards. Therefore, arbitrary-rate digital isochronous payloads originated by a customer cannot be transported over the existing public network. These payloads are digital signals synchronous to the user's application clock, having their own particular native format or structure, and having some rate that may not bear any relationship to rates of standard telecommunications signals.

Conceptually, to transport arbitrary-rate isochronous signals, a *virtual wire* is required. Such a service is transparent to the user, incurs very low constant delay, transfers clock synchronization at the application rate with minimum jitter, maintains the original bit rate and bit sequence, and has a very low bit-error-rate (BER). Such a service is not possible using current STM technology without requiring the customer to provide his own dedicated equipment that converts his signal into a standard format such as DS1, DS3, or STS-N. Transportation of isochronous payloads via ATM could require elaborate means to recover synchronization of the customer's signal (e.g. SFET), and the possibility exists for packet loss due to congestion at an ATM access node.

A new mode of transfer is necessary to accommodate customers having arbitrary-rate isochronous digital signals, providing them with a virtual wire service. UIPM can provide such a service by adapting these customer's signals into standard STM-based carrier signals. For each customer, UIPM appears as a dedicated system, but the same UIPM system can adapt any customer's signal.

Table 2.1 provides a comparison of attributes between STM, ATM, UIPM, and the concept of a virtual wire. The two main strengths of UIPM highlighted in Table 2.1 are its ability to maintain synchronization and accommodate signals of any rate. Neither STM nor ATM can claim both of these attributes. It can be seen from the comparison in Table 2.1 that UIPM comes the closest to offering a virtual wire transport.

Table 2.1 Comparison of attributes

Desired Attribute	Virtual Wire	STM	ATM	UIPM
lossless	Y	Y	N	Y
order preserving	Y	Y	Y	Y
low delay	Y	Y	N	Y
constant delay	Y	Y	N	Y
rate-insensitive	Y	***N***	Y	***Y***
maintains synchronization	Y	Y	***N***	***Y***
low BER	Y	Y	?	Y
low jitter	Y	Y	N	Y
bandwidth on demand	N/A	N	Y	N

Y = yes, N = no, ? = inconclusive, N/A = not applicable

The question mark in the "low BER" row for ATM indicates that the severity of error mechanisms in ATM systems is not yet known. Theses error mechanisms can arise from cell loss due to congestion and cell miss-routing due to errors in the HEC field.

A further comparison can be made between ATM and UIPM in terms of their basic purpose by considering each as a service adapter or transport interface. Both provide an interface to an STM-based transport mechanism (DS1, DS3, SONET) for user-based signals. This comparison is not meant to show superiority of one interface method over the other, but rather to provide a perspective of the relationship between ATM, UIPM, and STM. This relationship is summarized in Table 2.2 and illustrated in Figure 2.13.

Table 2.2 Comparison of ATM and UIPM as service adapters

Transport Interface:	ATM	UIPM
Transport Method: (physical layer)	SONET or native ATM	DS1, DS3, SONET
User Application:	multimedia, file transfer	arbitrary-rate digital isochronous payload transport

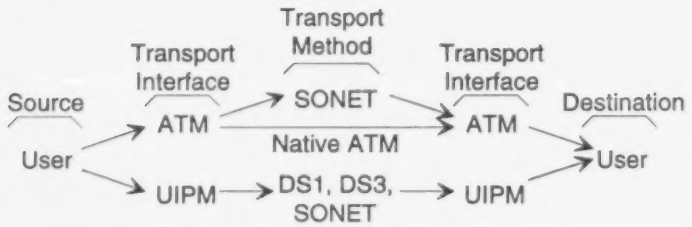


Figure 2.13 ATM and UIPM depicted as interfaces for user signals

2.6 UIPM SYSTEM CONCEPT

A UIPM system provides access to the public network for an arbitrary-rate digital isochronous signal by adapting it into the format of a standard transport carrier, and reversing the process at the destination. Figure 2.14 illustrates this idea in a pictorial fashion for one possible implementation of a UIPM system.

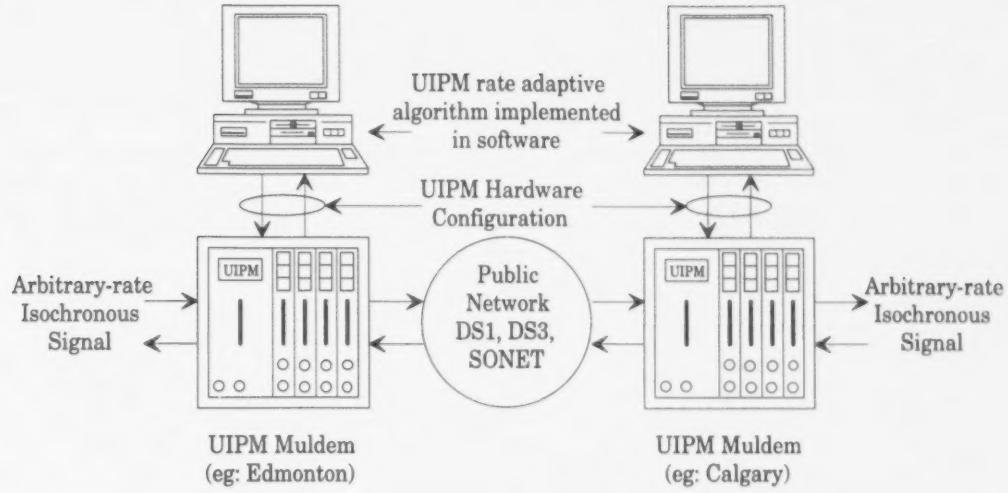


Figure 2.14 Implementation of UIPM (Muldem = Multiplexer/Demultiplexer)]

UIPM involves re-configurable multiplexer/demultiplexer hardware under the control of an arbitrary-rate multiplex design algorithm called the UIPM *format design algorithm* developed in this thesis. This algorithm is capable of optimum design of plesiochronous payload mapping formats for low waiting time jitter. All UIPM applications are specialized instances of plesiochronous multiplexing and use combinations of fixed and variable pulse stuffing justification. Waiting time jitter is therefore a central technical issue for this development.

The UIPM format design algorithm determines the length and structure of a variable-length UIPM frame format into which a customer's tributary signal is mapped, resulting in minimal jitter. Use of a variable-length plesiochronous frame format is a new concept. The resulting UIPM format is based on parameters of the customer's signal and on the desired standard carrier format used. The UIPM re-configurable hardware creates an output signal whose rate corresponds to the payload portion of the standard carrier signal used on the output side.

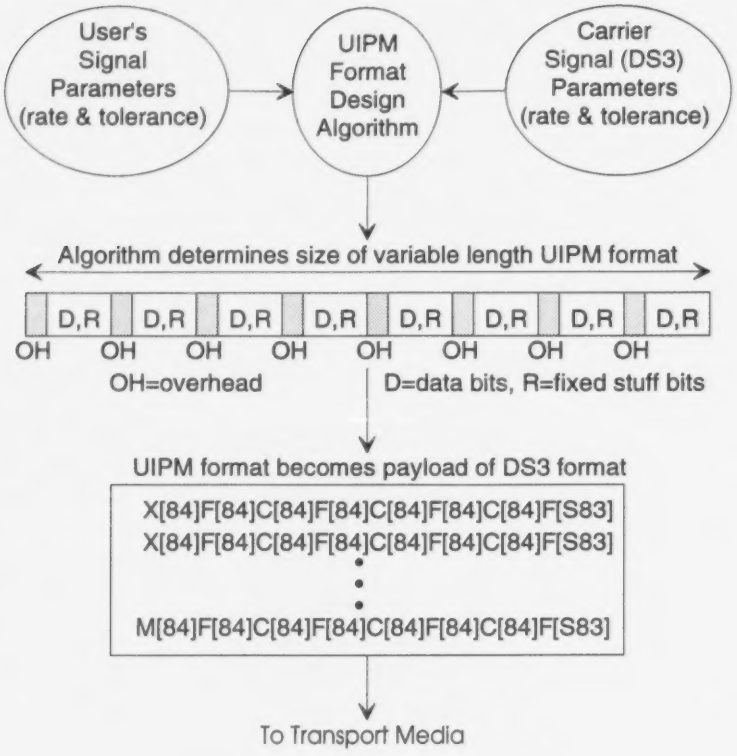


Figure 2.15 UIPM process

Figure 2.15 illustrates the UIPM process for adapting a user signal into the payload of a DS3 carrier. This procedure results in a customized plesiochronous multiplex format for each arbitrary-rate input signal. Additional UIPM hardware adds the conventional overhead structure to the UIPM format to create a fully standard signal for hand-off to the public network. In the DS3 example, this means adding M, C, P, X, and F overhead bits necessary for the DS3 format, resulting in the standard DS3 rate of 44.736 Mb/s. The rate and frame-level format of the DS3 carrier is thus retained at the network interface point.

Several stages of sophistication could be developed for UIPM technology, ranging from a system having one arbitrary-rate input and one standard rate output as described above, to a system having *n* mixed-rate inputs and *m* standard output carriers as shown in

Figure 2.16. In this most generalized configuration, UIPM is used with several tributaries per output carrier to create an efficient packing of the n inputs to m outputs ($m<n$). The "one-in/one-out" (1/1) scenario will be of primary concern in this study as this is the most central and technically fundamental aspect of all subsequent developments. Aspects of an n-in/m-out (n/m) UIPM system are discussed in Chapter 7.

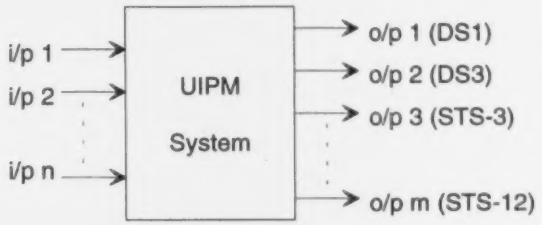


Figure 2.16 n/m UIPM system

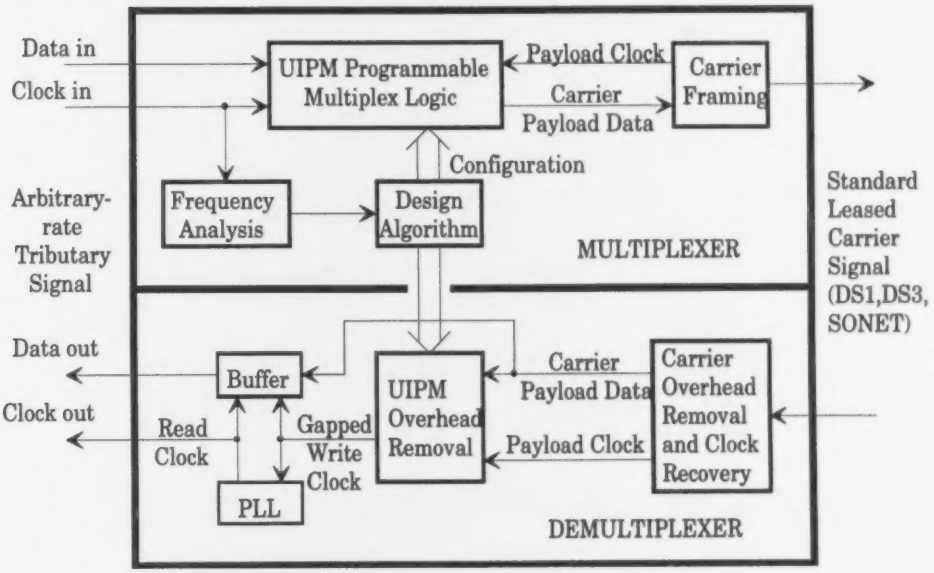


Figure 2.17 UIPM conceptual block diagram

A block diagram of a hypothetical 1/1 UIPM system is depicted in Figure 2.17. This system includes a UIPM format design algorithm and re-configurable multiplexing hardware consisting of multiplex/demultiplex circuitry, pulse stuffing/destuffing circuitry,

and associated desynchronizer PLL circuitry. The algorithm can be implemented in software running on an appropriate computer interfaced to the hardware circuitry. This method allows the UIPM system to handle any input/output combination; however, format design can be time consuming due to the iterative nature of the algorithm, and would not accommodate signals requiring bandwidth on demand. Alternatively, the algorithm can be implemented as a data base of suitably overlapping precomputed format structures residing in an EPROM directly imbedded in the UIPM hardware. A disadvantage of this method is that a large amount of memory could be required depending on the frequency steps desired at which designs are centered, but the time required to determine and configure a format design would be greatly reduced.

2.7 SCOPE AND CONTRIBUTION

There are three main UIPM components developed in this thesis: the variable length frame format, the UIPM format design algorithm, and experimental hardware. The prediction and measurement of jitter as the primary performance indicator is a central issue in this thesis. Emphasis is therefore placed on the development of the format design algorithm and the desynchronizer PLL, rather than construction of a complete UIPM prototype system, since these are the key technical enablers of UIPM technology and provide a means for predicting and measuring jitter performance.

The algorithm satisfies a number of design constraints such as the rate and tolerance specifications of the tributary and carrier, the maximum UIPM format size permitted, and jitter performance. In addition, there must be a fractional bit difference remaining between the number of tributary bits and the total bits in each UIPM frame. This fractional difference is accommodated by pulse stuffing justification. An iterative procedure is used to meet all design constraints until a UIPM format size is defined that exhibits minimum waiting time jitter. Jitter performance is simulated using techniques

from [MoKG91].

To facilitate jitter measurements on the PLL for specific UIPM format configurations, the functionality of the pulse stuffing mechanism in the multiplexer and clock gapping logic in the demultiplexer were combined in a single UIPM gapped-clock emulator circuit. Once a UIPM format was defined for a particular tributary-carrier combination, this information was used to determine parameters of the desynchronizer PLL and to manually configure programmable hardware in the clock gapping circuit. For proof-of-concept and performance measurements, a DS3 was selected as the carrier signal into which tributary signals of various rates were to be mapped. The actual jitter that would be imparted to a tributary signal, due to the configurable pulse-stuffing synchronization process in a real UIPM system, was experimentally measured.

2.8 DS3 PAYLOAD RATE

The concept of UIPM accommodates any tributary rate and any standard carrier rate. However, for this research, tributary-carrier combinations were limited to signals adapted into a DS3 payload. Since the DS3 format normally transports 7 DS2 tributaries, the nominal payload rate is $7(f_{DS2})$ = 7(6.312 Mb/s) = 44.184 Mb/s. This utilizes a percentage of the 7 stuff opportunities (S bits) existing in the DS3 format resulting in a nominal stuff ratio of 0.3906.

In a UIPM system, however, the DS3 carrier payload consists of the UIPM variable-length format, which in turn carries the tributary signal as its payload (see Figure 2.2). The bit rate of the UIPM format is equal and synchronous to the DS3 payload rate and therefore the seven DS3 stuff opportunities can all be used to carry DS3 payload data (equivalent to a normal DS3 format stuff ratio of 0). The C bits in the DS3 format must then be set to indicate that all the S bits in the DS3 format carry data. The plesiochronous nature of the tributary signal is handled by the pulse stuffing mechanism of UIPM. As a

result, no waiting time jitter is added by the DS3 variable stuff bits.

The target DS3 payload rate for UIPM format design, then, is the total number of available DS3 payload bits multiplied by the DS3 frame rate. The total available DS3 payload bits consists of the bits in the DS3 format dedicated for payload data (also called information bits) plus the 7 stuff opportunities or, equivalently, the total number of bits in a DS3 frame minus the number of fixed overhead bits. This results in 4704 of the 4760 bits in a DS3 frame available for payload. Since the DS3 frame rate is $f_f = f_{DS3}/(\text{total bits})$, the target carrier payload rate, rounded up to the nearest b/s, is:

$$f_p = 4704 \text{ bits}\left(\frac{44.736 \text{ Mb / s}}{4760 \text{ bits}}\right) = 44.209694 \text{ Mb / s}$$

2.9 SUMMARY

UIPM is intended to provide access to the public telecommunications network for arbitrary-rate digital isochronous signals. Such a service cannot be provided by existing transport methods while developing standards such as ATM pose other problems associated with asynchronous communications. A complete UIPM prototype system maps nonstandard digital signals into the payload of a standard synchronous carrier by using re-configurable multiplexer/demultiplexer hardware under the control of a UIPM format design algorithm. A key issue for the UIPM format design algorithm is control of waiting time jitter while satisfying a number of other constraints.

The main scope of this thesis involves the development of UIPM components for the case where a single input is adapted into the payload portion of a single DS3 carrier. By itself, this will be inefficient at low tributary rates but it serves the research objective of validating the UIPM concept. Multiple tributary-carrier combinations can then be engineered to suit the actual economic requirements for loading a leased DS3 to a suitable level.

The experimental hardware (detailed in Chapter5) consists of a UIPM

desynchronizer PLL used for recovery of the original tributary clock, and a clock gapping circuit that emulates time-base behavior of a UIPM system. The experimental hardware is sufficient for providing proof-of-concept results.

A detailed description of the UIPM components developed commences with the next chapter which discusses the function of format standards for signal transport followed by a description of the variable length frame format used in UIPM.

3 UIPM GENERIC FORMAT DESCRIPTION

This chapter discusses the variable length format structure used in a UIPM system to map an arbitrary-rate isochronous signal into the payload portion of a standard carrier signal such as a DS3 or an STS-3. The general function and structure of formats used for signal mapping purposes is discussed followed by a detailed description of the UIPM variable length format (called the *generic format*) including limits imposed on this format in terms of size. For purposes of the thesis the standard carrier used is a DS3. A discussion is thus given on the bandwidth utilization of the DS3 payload by a UIPM generated signal.

3.1 BASIC PRINCIPLES

The UIPM variable length format uses concepts similar to those in plesiochronous multiplexing, previously discussed. The following describes plesiochronous multiplexing in general terms and compares it to the method developed for UIPM for transport of arbitrary-rate isochronous digital signals.

To convert a tributary signal into a higher rate standard carrier, extra bits must be added to the original signal at the transmit end. However, these extra bits cannot be added in a random fashion but must be multiplexed with the original bit stream such that they can be identified and removed at the receive end. The way this occurs is dictated by a format that describes the purpose of each bit time slot within a repetitive frame of total bits. Each frame occurs in a certain time period resulting in the desired bit rate. The extra bits are used for the following: 1) to increase the data rate (by using fixed stuffing); 2) to convey information for framing, stuff bit status, and perhaps error detection or correction; 3) for variable bit stuffing to accommodate the tolerances of the tributary and carrier rates.

The types of bits included are:

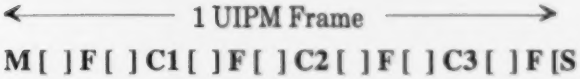
- the payload data bits (or information bits) from the tributary signal
- fixed stuff bits, if necessary, to increase the rate by a fixed amount
- variable stuff bits to account for frequency offsets in tributary and carrier signal rates and to accommodate fractional number of tributary bits per carrier frame
- stuff indication bits to provide status of the variable stuff bits
- additional overhead bits for error detection and correction or other purposes
- framing bits to allow receiver alignment on the signal format

Examples of formats similar to that described above are the DSn and SONET formats discussed in Chapter 2. All such formats used today, share one additional characteristic - they are all fixed length formats in that they consist of a fixed number of bits in a frame. This limits the range of tributary rates that can be accommodated by these formats, and prohibits the transport of arbitrary-rate payloads.

The UIPM format, on the other hand, is a rate-adaptive variable length format whose frame length can change to accommodate a wide range of tributary rates. The basic structure of the UIPM format remains fixed while the number of data and fixed stuff bits varies according to an algorithm that determines an optimum frame length based on a number of design objectives. Since the same basic format structure is used for any tributary rate, it is called the UIPM *generic format*. The UIPM generic format provides each payload rate with a unique customized format optimized for the characteristics of that particular payload. The variable frame length feature of the UIPM generic format is the key to transporting arbitrary-rate tributaries and accommodating the frequency offset ranges of the tributary and carrier.

3.2 DETAILED DESCRIPTION

The structure of the generic format frame is shown in Figure 3.1.



M = UIPM multiframe bit
F = UIPM framing bit
C1, C2, C3 = triplicated UIPM stuff indication bits
[ ] = non-stuffing payload subframe composed of: **[O_m, O_d, O_r, E, {nD, kR}]**
[S] = stuffing subframe composed of: **[O_m, O_d, O_r, E, S, {(n-1)D, kR}]**
where: **O_m** = overhead bit (future use)
O_d, O_r = subframe overhead bits: indicate distribution of D and R bits within subframe
E = error detection/correction bit
S = variable stuff bit
{nD, kR} = data element
{(n-1)D, kR} = data/stuffing element
D = payload data bit
R = fixed (reserved) stuff bit
n and **k** are integers determined by the UIPM format design algorithm

Figure 3.1 Generic Format structure

Each frame of the generic format consists of eight subframes preceded by a dedicated UIPM overhead bit (M, F, or C bit). Each subframe consists of dedicated overhead bits followed by an element containing data and fixed stuff bits distributed in a particular pattern. The last subframe contains a variable stuff bit (S) which carries either data or a dummy bit, as indicated by the triplicated UIPM C bits.

Within one frame of the generic format, there are: i) 7 data elements and one data/stuffing elements; ii) 40 dedicated overhead bits consisting of the M bit, 4 F bits, 3 C

bits, and 8 sets of the O_m, O_d, O_r, and E bits (in that order); iii) one variable stuff bit, S. Bits are read from left to right starting with the M bit to form a bit stream whose rate corresponds to the payload portion of the standard carrier to be used. The total number of bits per frame depends on the tributary and carrier payload rates, their respective tolerances, and other design criteria including minimum simulated waiting time jitter.

The F bits are used for framing on the UIPM generic format in the demultiplexer portion of a UIPM system. The M bit is for future use for alignment of multi-tributary UIPM frames. Initially, neither the frame boundaries nor the UIPM frame size are known at the demultipexer. In conventional framing, only the framing bit position is unknown and frame alignment is established by searching for a framing bit pattern. For the UIPM generic format, information about the frame size could be conveyed to the demultiplexer manually (i.e. as installation/setup data), or the frame boundaries and frame size could be determined through a "two dimensional" search procedure. In the later case the demultiplexer would initially assume the minimum possible frame size, search for the framing pattern, then increment the frame size if framing cannot be established and resume the search until the framing pattern is found. Section 3.5 provides an analysis of the increase in initial framing time for the two dimensional case, where the frame size is not initially known, over the conventional case ("one dimemsional") where the frame size is known.

Once frame alignment is accomplished, the position of all other bits is known. The O_d and O_r overhead bits, which together form a 16 bit word every frame, indicate the pattern of data and fixed stuff bits common to all subframes. The C bits indicate, by majority vote, whether the S bit in the given UIPM frame carries a data bit or a dummy bit, i.e. whether or not the UIPM tributary synchronizer has performed a stuffing operation for this frame. While in frame, the UIPM demultiplexer would continually execute a confirmation process to verify that frame alignment is maintained. Figure 3.2 summarizes the framing and decoding procedure in the form of a flow diagram.

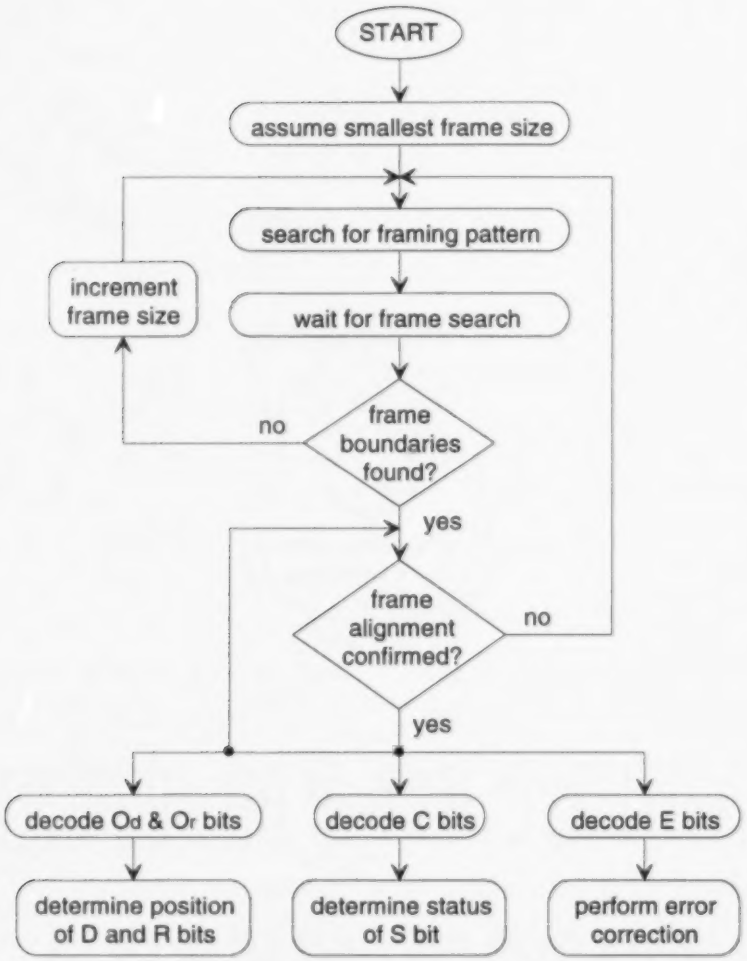


Figure 3.2 Framing and decoding process

3.3 SIZE CONSTRAINTS

The maximum size of the UIPM subframe was set at 512 bits resulting in a maximum frame size of 4104 bits. This provides a reasonable limit to the size of dividers and counters required in UIPM hardware. In UIPM format design, the frame size can be incremented by 8 bits at a time by adding one D bit per element. Table 3.1 summarizes the

size constraint characteristics of the generic format in terms of the number of bits per subframe and per frame for each type of bit used.

Table 3.1 Fixed and variable dimensions of the UIPM generic format

BITS	PER SUBFRAME	PER FRAME
M	-	1
F	-	4
C	-	3
S	-	1
O_m	1	8
O_d	1	8
O_r	1	8
E	1	8
nD	n=1,2,3,...,508	8n-1
kR	k=0,1,2,...,507	8k
total bits	t=5,6,7,...,512	8t+8

The minimum and maximum frame sizes are 48 and 4104 bits respectively while the minimum size increment is 8 bits. The minimum number of data bits per subframe is 1 rather than 0 because no information would be conveyed with 0 data bits. The minimum number of data bits per frame is 7 rather than 8 since the S bit may be used for stuffing rather than for data. The minimum number of R bits, however, can be 0 without violating any design criteria in the UIPM format design algorithm.

The only component of the generic format that can vary in size is the element which contains the data and fixed stuff bits. The element size can increment by one bit and the maximum number of bits in an element is 508. Therefore, in UIPM format design, the element size needs to be monitored to determine if the generic format has reached its maximum size. Because the number of data bits and fixed stuff bits are the only variable

components of the UIPM generic format, the integers n and k completely define a given UIPM format design in terms of size.

3.4 OPERATING DYNAMIC RANGE

The operating dynamic range of a UIPM signal carried as the payload in a DS3 frame is the range of DS3 bandwidth used by the UIPM overhead and UIPM payload (tributary data). This range exists between the minimum and maximum UIPM format sizes.

Figure 3.3 shows the frame structure for the conventional DS3 signal. The X, P, and M bits are master frame overhead bits; the F bits are used for framing; the C bits indicate the status of the variable stuff (S) bits for each of the 7 DS2 tributaries; and the D bits are DS3 payload data bits. The sequence of bits in a bit stream is from left to right and top to bottom.

X [84D] F_1 [84D] C_{11} [84D] F_0 [84D] C_{12} [84D] F_0 [84D] C_{13} [84D] F_1 [0D S_1 83D]
X [84D] F_1 [84D] C_{21} [84D] F_0 [84D] C_{22} [84D] F_0 [84D] C_{23} [84D] F_1 [1D S_2 82D]
P [84D] F_1 [84D] C_{31} [84D] F_0 [84D] C_{32} [84D] F_0 [84D] C_{33} [84D] F_1 [2D S_3 81D]
P [84D] F_1 [84D] C_{41} [84D] F_0 [84D] C_{42} [84D] F_0 [84D] C_{43} [84D] F_1 [3D S_4 80D]
M_0 [84D] F_1 [84D] C_{51} [84D] F_0 [84D] C_{52} [84D] F_0 [84D] C_{53} [84D] F_1 [4D S_5 79D]
M_1 [84D] F_1 [84D] C_{61} [84D] F_0 [84D] C_{62} [84D] F_0 [84D] C_{63} [84D] F_1 [5D S_6 78D]
M_0 [84D] F_1 [84D] C_{71} [84D] F_0 [84D] C_{72} [84D] F_0 [84D] C_{73} [84D] F_1 [6D S_7 77D]

Figure 3.3 DS3 frame structure [Ansi88b]

In one DS3 frame there are a total of 4760 bits of which 56 bits (8 DS3 overhead bits for each of the 7 DS3 frames) are used for DS3 overhead (X, P, M, F, and C bits) and the remaining 4704 bits (including the 7 stuff bits) are available for DS3 payload data. Of the total available bandwidth in one DS3 frame the amount consumed by the DS3 format

overhead bits is $\frac{56}{4760}(100) = 1.18\%$ which leaves 98.82% for the DS3 payload. For purposes of the thesis, the DS3 payload will consist entirely of the UIPM signal.

The maximum UIPM payload capacity will occur when the UIPM generic format is at its maximum frame size and no fixed stuff (R) bits are used. In this case, the UIPM overhead consists of 40 bits per UIPM frame (not including the UIPM format's S bit) with a UIPM frame size of 4104 bits. Therefore, in one DS3 frame there are 4104 UIPM format bits (one entire UIPM frame) plus:

(4704 DS3 payload data bits)-(4104 UIPM format bits) = 600 UIPM bits remaining.

Of these 600 bits the first 5 are UIPM overhead (OH) bits, the next 508 bits are UIPM customer payload bits (not to be confused with DS3 payload bits), the next 5 are again OH bits, and the remaining 82 are UIPM payload bits. Thus in one DS3 frame there are:

$$40+5+5 = 50 \text{ UIPM OH bits}$$

and (4704 DS3 payload data bits)-(50 UIPM OH bits) = 4654 UIPM payload bits.

For a maximum generic format size then, the amount of DS3 bandwidth consumed by UIPM overhead is:

$$\frac{50 \text{ UPM OH bits}}{4760 \text{ total DS3 bits}}(100) = 1.05\%$$

and that of UIPM payload is:

$$\frac{4654 \text{ UIPM payload bits}}{4760 \text{ total DS3 bits}}(100) = 97.77\%.$$

The minimum UIPM payload capacity will occur when the UIPM generic format is at its minimum frame size of 48 bits. In this case there are 0 UIPM fixed stuff bits and 1 data bit per element, or 8 data bits per UIPM frame (this includes the one S bit in the last subframe). This results in:

$$\frac{4704 \text{ DS3 payload bits}}{48 \text{ UIPM bits / frame}} = 98 \text{ UIPM frames in one DS3 frame.}$$

The amount of DS3 bandwidth consumed by essential UIPM overhead is then:

$$\frac{(98 \text{ UIPM frames})(40 \text{ UIPM OH bits / frame})}{4760 \text{ total DS3 bits}}(100) = 82.35\%$$

and that of UIPM payload is:

$$\frac{(98 \text{ UIPM frames})(8 \text{ UIPM payload bits / frame})}{4760 \text{ total DS3 bits}}(100) = 16.47\%.$$

It is obvious that the minimum generic format size does not represent a very efficient utilization of DS3 bandwidth. However, the maximum and minimum format sizes are bounds within which the generic format structure can operate. The issue of efficiency is further addressed in Chapter 4. Figure 3.4 summarizes the variable bandwidth utilization range of a UIPM signal carried as the payload in a DS3 frame.

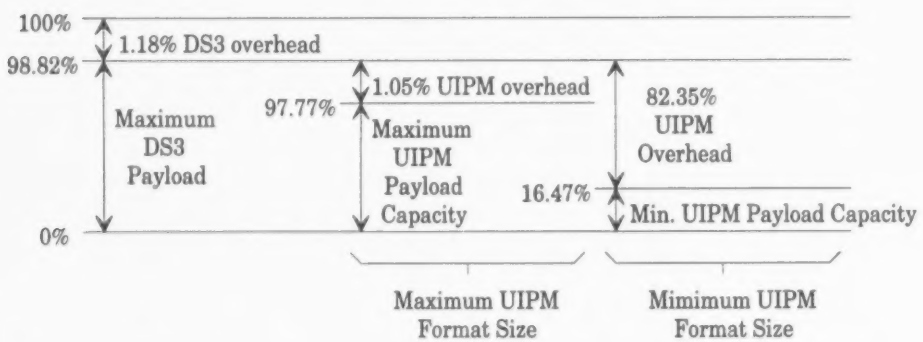


Figure 3.4 Variable bandwidth utilization by a UIPM signal

3.5 FRAMING TIME FOR TWO DIMENSIONAL SEARCH

To determine the time required for initial framing using a two dimensional search procedure, where the frame size is unknown, first consider the conventional framing procedure for a fixed format of known frame size. Let L-1 be the number of bits between framing bits and define L as the framing interval length (in bits). The time period of one bit is $1/f_b$ where f_b is the bit rate. Then the framing interval duration is L/f_b seconds. The longest search time will result when the first bit tested occurs just after a framing bit. This

represents a maximum length search and takes L-1 tries which requires $L(L-1)/f_b$ seconds. However, after each unsuccessful try, one bit period will elapse before the next position is tried. For a maximum length search this adds an additional $(L-1)/f_b$ seconds.

For binary data, there is a 50% chance that the bit being detected is a data bit mimicking a valid framing bit. This implies that on average, the framing search will stay at each search position two times which doubles the search time. However, the actual starting position of the search is random so that on average, the search will start half way between framing bits which reduces the search time by a half. Therefore, the average maximum length search time for conventional framing is:

$$t_{ave\ max} = \frac{L(L-1)}{f_b} + \frac{L-1}{f_b} = \frac{L^2-1}{f_b} \tag{3-1}$$

For the two dimensional search the procedure initially assumes the minimum framing interval length (L_{min}). Let ℓ represent the current framing interval length for which frame alignment is being sought, and let L be the true framing length interval. All searches will be unsuccessful for $L_{min} \leq \ell < L$. For all unsuccessful searches, the notion that a search will start half way between framing bits is meaningless because the assumed framing interval length is not the correct value. However, the search will on average remain at each position two times. Therefore the average maximum length search time required for each unsuccessful search will be $2\ell(\ell-1)/f_b+(\ell-1)/f_b = (2\ell^2-\ell-1)/f_b$. After this amount of time elapses, an unsuccessful search will continue forever unless a *search fail time* (SFT) limit is imposed for which there is a high degree of certainty framing cannot be achieved. This time limit can be stated as $k\sigma\ell/f_b$ where k is an arbitrary factor and σ is the standard deviation of the probability density function (pdf) for the maximum length search time illustrated in Figure 3.5.

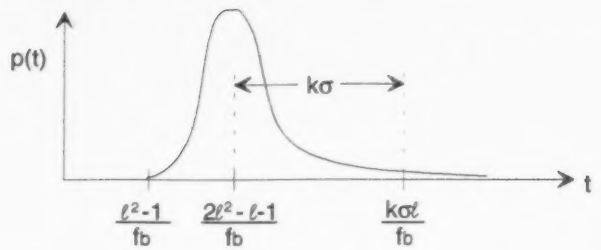


Figure 3.5 pdf of an unsuccessful maximum length search

The pdf shown in Figure 3.5 does not represent any standard pdf because the actual pdf is unknown. However, it will be asymmetrical because the minimum time required for a maximum length search is $\ell(\ell\text{-}1)/f_b+(\ell\text{-}1)/f_b = (\ell^2\text{-}1)/f_b$ (same as for conventional framing), for which no data bits mimic valid framing bits, while the maximum time required is unbounded for unsuccessful searches.

The total average maximum length search time required for the two dimensional search is the sum of all unsuccessful searches plus the last search where ℓ=L, which will be successful. This can be stated as follows:

$$t_{\text{ave max 2D}} = \sum_{\ell=L_{\min}}^{L-1} \left(\frac{2\ell^2 - \ell - 1}{f_b} + k\sigma \cdot \frac{\ell}{f_b} \right) + \frac{L^2 - 1}{f_b} \qquad (3\text{-}2)$$

A comparison can be made between the two dimensional and conventional average maximum length search times by dividing equation 3-2 with equation 3-1. This gives the increase in time (t_{inc}) two dimensional framing will take over conventional framing:

$$t_{inc} = \sum_{\ell=L_{\min}}^{L-1} \left(\frac{2\ell^2 - \ell - 1 + k\sigma\ell}{L^2 - 1} \right) + 1 \qquad (3\text{-}3)$$

To get a feeling for what the increase in framing time will be numerically, equation 3-3 is evaluated for three values of the SFT ($k\sigma\ell/f_b$):

a) SFT = the average maximum length search time - then $k\sigma\ell = 2\ell^2\text{-}\ell\text{-}1$;

b) SFT = twice the average maximum length search time - then $k\sigma\ell = 2(2\ell^2\text{-}\ell\text{-}1)$;

c) SFT = three times the average maximum length search time - then $k\sigma\ell = 3(2\ell^2\text{-}\ell\text{-}1)$.

Figure 3.6 shows a log-log plot of t_{inc} versus the true framing interval length (L) as L varies from L_{min} (6 bits) to L_{max} (513 bits) for each value of SFT. For L>10, the relationship between t_{inc} and L is essentially linear. It should be noted that the increase in framing time represents initial framing only during commissioning of the transport link. If the network experiences a disturbance that causes UIPM equipment to loose frame alignment, then the framing procedure will assume the last known framing interval length. Hence, reframes in this sense will take the same time as conventional framing.

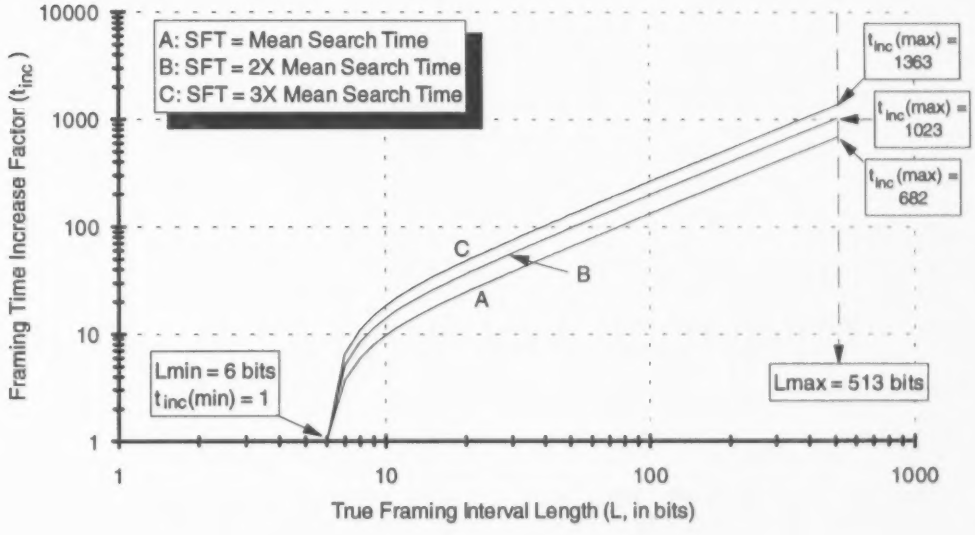


Figure 3.6 Increase in initial framing time for two dimensional framing

3.6 SUMMARY

The UIPM generic format is a variable length structure used to map arbitrary-rate digital signals into the payload of a standard carrier. The detailed form of the generic format structure is determined by the UIPM format design algorithm for any given tributary rate.

Each frame of the generic format consists of 8 subframes preceded by a dedicated overhead bit. Each subframe contains additional format-describing overhead followed by a data element that carries customer data and any necessary fixed stuff bits. The last subframe is of the same length but contains a variable stuff opportunity to accommodate frequency offsets and the non harmonic relationship between the tributary and carrier frame times.

The generic format structure can operate between minimum and maximum size constraints. Between these boundaries the utilization of DS3 bandwidth can be varied widely to accommodate arbitrary-rate tributaries. The variance in efficiency within this range is partially due to the limitations of the 1in/1out scenario adopted in the thesis. Efficient use of the available DS3 payload capacity can be accomplished by combining several identical tributaries into a higher rate signal before multiplexed into a UIPM format.

If a two dimensional search is used for framing, the increase in initial framing time (t_{inc}), compared to conventional framing, varies proportionally to the true framing interval length (L) for values of L>10. The constant of proportionality depends on the value chosen for the search fail time used to determine when an unsuccessful search should be abandoned.

Chapter 4 continues the detailed description of UIPM by discussing the development of the UIPM format design algorithm which determines the optimum size of the generic format for a given tributary/carrier combination.

4 UIPM FORMAT DESIGN ALGORITHM

This chapter presents the format design algorithm used in the UIPM system. The basic principle and functionality of the algorithm is discussed followed by specifics on the criteria used to determine feasible format sizes of the generic format structure. A section on the software implementation of the algorithm describes the general procedure used in the source code and then presents the procedure in terms of pseudocode and a flowchart. A discussion is then given on jitter simulation as it is used in the UIPM format design algorithm to determine the optimum format size from a set of feasible format designs. Finally, a brief description is given on other simulation tools used for format analysis.

4.1 BASIC PRINCIPLE

The UIPM format design algorithm determines the integers n and k of the generic format through an iterative procedure. This procedure selects one of all feasible format sizes meeting a set of design criteria based on characteristics of the tributary and carrier signals and on the size constraints of the generic format. The format size that gives the lowest simulated jitter is then selected for use. In a UIPM system the parameters of the selected generic format are passed to configurable hardware that performs the actual mapping process. The UIPM format design algorithm is a *rate adaptive* mapping algorithm because the iterative procedure attempts to find a final format structure for each combination of tributary and carrier rates and frequency tolerances presented to it.

The next section details the design criteria used by the algorithm to determine all feasible formats for a given set of input parameters. The decision criterion used to pick one of these possible formats is jitter performance. Absent from this decision process is the efficiency of the format, alluded to in Chapter 3. For efficient use of carrier payload capacity, the larger the UIPM format the better. However, efficiency is fundamentally defined by the tributary rate - a large UIPM format size may use a small percentage of

UIPM overhead compared to UIPM payload capacity, but if a low rate tributary is being transported, much of the UIPM payload capacity will be filled with fixed stuff (R) bits. In this case, efficiency is more properly defined as the percentage of total bits in a UIPM format frame dedicated for carriage of tributary data. For the 1/1 scenarios adopted in this thesis, cases where a low rate tributary is adapted into a DS3 payload are inherently inefficient. A practical realization of UIPM would eventually accommodate multiple arbitrary-rate tributaries and multiple standard carriers for the purpose of efficient bandwidth utilization. In this case a more complex UIPM algorithm would decide which tributaries to multiplex together and which carrier to use, resulting in overall maximum efficiency.

4.2 DESIGN CRITERIA

The following are criteria obeyed by the iterative process in the UIPM format design algorithm:

1. There must be an integral number of data bits per data element (n) (i.e. $n \in I$). The value of n is equal to the total number of data bits (d) in a frame, plus the S bit, divided by 8 since there are 7 data elements and one data/stuffing element per frame. Equation 4-1 summarizes this relationship:

$$n = \frac{d+1}{8},\ n \in \{1,2,3,\ldots,508\} \tag{4-1}$$

The total number of data bits per frame (d) is defined by:

$$d = \mathrm{Int}\left[\frac{f_t}{f_p} \cdot T\right] \tag{4-2}$$

where f_t is the tributary rate, f_p is the carrier payload rate, T is the total number of bits in a UIPM frame, and Int[x] is the integer part of [x]. Expressing T as the sum of

dedicated overhead bits, data bits, and fixed stuff bits, and combining equations 4-1 and 4-2, this criterion can also be expressed by stating that all combinations of n and k satisfying the following inequalities generate feasible UIPM formats:

$$8n - 1 \le \frac{f_t}{f_p}(40 + 8n + 8k) < 8n$$
$$\text{for all } n = 1,2,3,\ldots,508 \qquad (4\text{-}3)$$
$$\text{for all } k = 0,1,2,\ldots,507$$

2. Each feasible UIPM format must have the capacity to carry the total number of data bits per frame defined in the first criterion. In other words, for a given format size under consideration, the total number of bits available in a data element (eltot), due to the latest increment in size, must be greater than or equal to the number of data bits per data element (n) arrived at from Equation 4-1 (i.e. eltot≥n). These two amounts are determined independently of each other. Format sizes that cannot comply with this requirement are discarded.

3. There must be at least one data bit per element to convey payload information and to accommodate frequency offset tolerances of the tributary and carrier via the S bit.

4. The range of tributary frequency offset that can be accommodated by the format's fixed and variable stuffing capacity must be greater than or equal to the combined tributary and carrier offset range. This requires that the combined equivalent offset range of the tributary and carrier correspond to stuff ratios between zero and one. This criterion is described by equations 4-4 and 4-5 and by Figure 4.1, where f_f is the UIPM frame rate, and Δ is the combined tributary and carrier tolerance in ppm.

$$df_f \le f_t - f_t\Delta \quad (\rho \le 1) \qquad (4\text{-}4)$$

$$(d+1)f_f \ge f_t + f_t\Delta \quad (\rho \ge 0) \qquad (4\text{-}5)$$

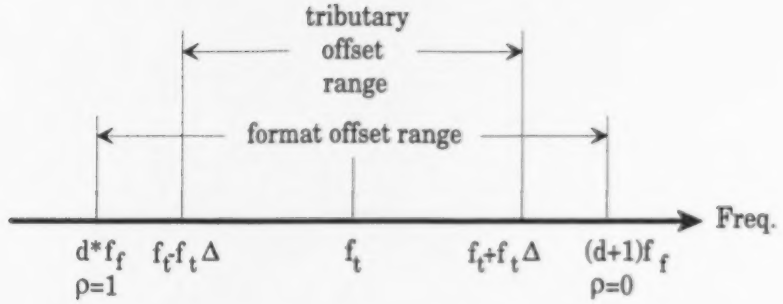


Figure 4.1 Combined tributary-carrier frequency tolerance criterion

5. The size constraints of the UIPM generic format, detailed in Chapter 3, form part of the design criteria for the UIPM format design algorithm by setting size and increment boundaries.

4.3 SOFTWARE IMPLEMENTATION

The UIPM format design algorithm was implemented in Pascal. Early versions were written in Turbo Pascal on an IBM compatible computer while the final version was compiled by a UNIX based Pascal compiler on a SUN Sparc station to enhance the execution time. The executable program was run on a number of workstations simultaneously resulting in an effective computing speed of 120 to 140 MIPS. Even with this amount of computing power a set of predicted jitter results of optimal UIPM formats for tributary rates ranging from 1 Mb/s to 44 Mb/s, in 10 kb/s increments, took about 3 weeks to complete.

In a realistic situation, the UIPM format design algorithm may run on a typical PC for which execution time to determine a UIPM format for a single tributary rate may take about 20 minutes. This setup time would not service signals requiring bandwidth-on-demand but would accommodate medium or long term connections. As discussed in

Chapter 2, a large data base of optimum UIPM format parameters for a range of tributary rates could allow a much faster setup time.

The pseudocode and flowchart sections describe the iterative procedure and the design criteria used in the algorithm. Details on the simulation of jitter used in the algorithm are given in Section 4.4. Appendix B provides documented source code which includes the main UIPM format design algorithm, the jitter simulation module, and an extra iterative loop used to capture predicted jitter of optimum UIPM formats for a range of tributary rates.

4.3.1 General Description

The primary inputs for the algorithm are the tributary and carrier rates and their respective frequency tolerances in ppm. Other inputs include parameters used for jitter simulation such as characteristics of the desynchronizer PLL. Outputs of the algorithm include parameters describing the final structure of the generic format, its nominal stuff ratio, and the maximum predicted jitter occurring within the combined tributary and carrier offset range in 10 Hz steps.

The generic format structure parameters consist primarily of the number of D and R bits per frame (i.e. n and k). The value of the O_d and O_r bytes are also required to define the distribution of D and R bits within each element. However, the present version of the algorithm does not determine the value of the O_d and O_r bytes because the experimental hardware does not include the data path of a complete UIPM system. For the clock gapper circuit of the experimental hardware, the distribution of D bits (resulting in no gaps) and R bits (resulting in gaps) is determined by inspection and entered manually via dip switches.

The main iteration procedure of the UIPM format design algorithm searches for all formats that will satisfy the design criteria by starting with a format of the minimum size and incrementing the size by one byte until a feasible format is found. The first feasible

format found corresponds to frame length (or size) called level one. The iteration procedure is then continued by incrementing the format size until the next feasible format is found referred to as the level two frame length. This continues until the design reaches its maximum frame size restriction. Each time a feasible format is found its jitter performance is simulated and compared with the previous format. For each feasible format, the jitter simulation determines the largest peak-to-peak (p-p) jitter (at the output of the desynchronizer PLL) occurring anywhere in the combined frequency offset range of the tributary and carrier rates (in 10 Hz steps). The format exhibiting the lowest p-p simulated jitter is the one chosen as optimum for that particular design case.

Jitter can be expressed in terms of p-p or rms. Specifications for jitter performance of standardized mapping formats often provide limits for both p-p and rms jitter after the PLL which acts as a low pass filter for phase. Jitter resulting from the mapping or synchronization process is referred to as "wideband" (WB) jitter while the jitter remaining after the desynchronizer PLL is termed "narrowband" (NB) jitter. Unless otherwise stated, any discussion on jitter in this thesis refers to NB jitter.

The p-p specification is an absolute limit regardless of the signal statistics in terms of phase noise. The rms specification, however, may be based on an assumed probability density function (pdf) of the narrowband jitter, whether or not the pdf truly describes the jitter characteristics of the desynchronized signal. For the UIPM generic format the pdf of the jitter may well differ from case to case. Thus it was not initially known which specification of jitter to use (p-p or rms or both) for determining a final structure with lowest jitter. However, after preliminary simulation and analysis, it was determined that comparison of p-p jitter alone provided the best criteria for lowest overall p-p and rms jitter.

Figures 4.2 and 4.3 show the difference between predicted UIPM format performance based on p-p jitter as opposed to rms jitter. The simulation details were identical for both sets of results and a discussion of them is not relevant at this point. It is

clear that UIPM formats based on predicted p-p jitter performance have considerably lower jitter than formats based on predicted rms jitter performance.

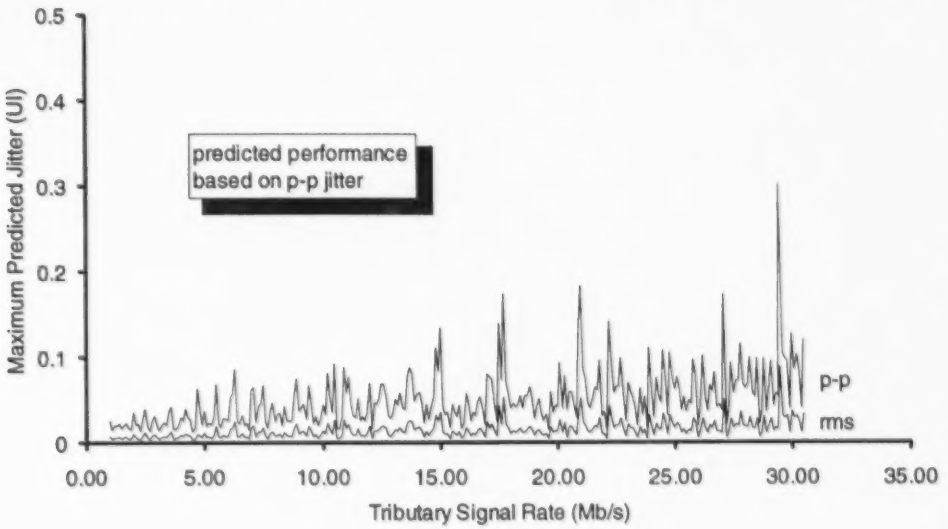


Figure 4.2 Predicted jitter performance of UIPM formats based on p-p jitter

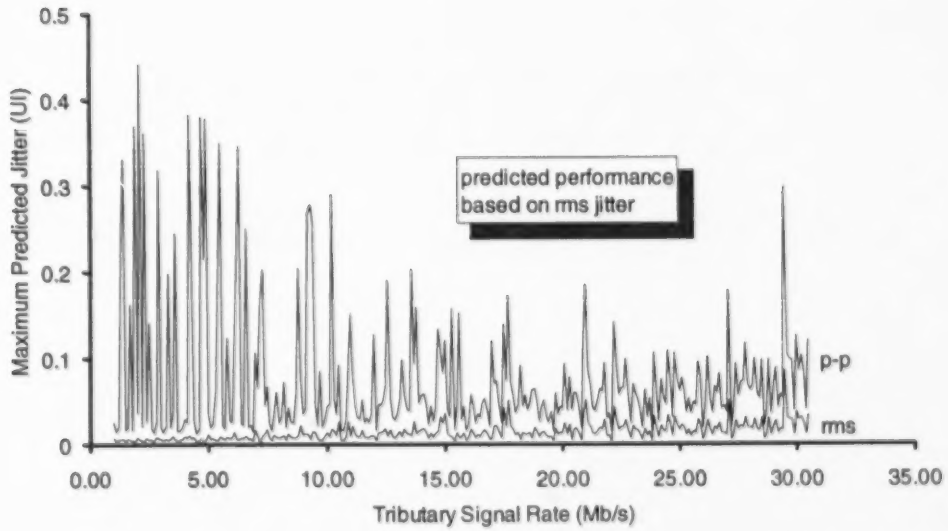


Figure 4.3 Predicted jitter performance of UIPM formats based on rms jitter

For any given tributary and carrier rate combination there may be a few or many feasible designs generated by the algorithm before the maximum size is reached. Because of the size increment limitation of one byte, there may be cases where a feasible format cannot be found at all before the maximum size is reached. The jitter simulation process requires the greatest amount of CPU time compared to the rest of the UIPM algorithm because it scans the entire frequency offset range for each feasible format design. Therefore there is a significant increase in execution time for designs containing many levels or feasible formats.

The following example shows how the set of design criteria is used to determine feasible formats. Table 4.1 shows intermediate results of the UIPM format design algorithm for a 10 Mb/s tributary adapted into a DS3 carrier payload. The combined tributary-carrier frequency tolerance is +/-100 ppm resulting in a frequency offset range of 2 kHz, or a tributary range from 9.999000 Mb/s to 10.001000 Mb/s with a carrier tolerance of 0. There are 57 feasible formats for the 10 Mb/s case but only searches for levels 1, 2, and 43 format sizes are given. A check mark in the first column indicates feasible formats.

Table 4.1 shows the total number of bits in a data element (eltot) as the format size increments by 1 bit per subframe, or 1 byte per frame, and the corresponding value for n and the available offset range. For each level, the total number of bits per frame (T), the total number of D bits per frame (d), the number of R bits per subframe (k), and the predicted jitter performance, is given. For this example, the level 1 format size exhibits the lowest waiting time jitter.

The last column in Table 4.1 shows the criteria violated by non-feasible format sizes. The primary criteria of concern are #1: frac(n)=0, #2: eltot≥n, and #4: adequate format offset range. As the generic format increments in size, criterion #1 is the one most often violated. An interesting non-feasible format size is the one for eltot = 357 bits.

Here criterion #1 is not violated (n = 82.000) but criterion #4 is - there is not adequate offset range for the tributary.

Table 4.1 Intermediate results of UIPM format design algorithm for 10 Mb/s case

eltot (eltot≥n)	n (frac[n]=0)	Available Offset Range (Hz)		Criterion violated
		$f_t$min	$f_t$max	
1	1.375	9210352.941167	10131388.235283	1&2
2	1.625	9473505.882343	10262964.705871	1
3	1.875	9670870.588225	10361647.058813	1
4	2.125	9824376.470578	10438399.999989	1
5	2.375	9947181.176460	10499802.352930	1
6	2.500	9545274.866300	10047657.754000	1
7	2.750	9670870.588225	10131388.235283	1
√ 8	3.000	9777143.891392	10202237.104062	none
level = 1, T = 104, d = 23, k = 5, jitter = 0.0158 UI rms, 0.0717 UI p-p_{max}				
9	3.250	9868235.294107	10262964.705871	1
10	3.500	9947181.176460	10315595.294107	1
11	3.625	9670870.588225	10016258.823519	1
12	3.875	9752138.408294	10077209.688571	1
13	4.125	9824376.470578	10131388.235283	1
14	4.375	9889010.526305	10179863.777079	1
15	4.625	9947181.176460	10223491.764695	1
16	4.875	9999811.764695	10262964.705871	1&4
√ 17	5.000	9796466.310150	10047657.754000	none
level = 2, T = 176, d = 39, jitter = 0.0194 UI rms, 0.1278 UI p-p_{max}				
354	81.250	9990282.549555	10005675.897089	1
355	81.500	9993232.941166	10008583.529401	1
356	81.750	9996166.987117	10011475.052947	1
357	82.000	9999084.822869	10014350.601224	4
358	82.125	9986762.858521	10001986.582391	1
359	82.375	9989690.497727	10004872.398179	1
360	82.625	9992602.095074	10007742.401279	1
361	82.875	9995497.782053	10010596.721301	1
362	83.125	9998377.688722	10013435.486446	1
363	83.250	9986225.063928	10001241.943723	1
364	83.500	9989114.490664	10004090.674308	1
365	83.750	9991988.298876	10006924.006349	1
√ 366	84.000	9994846.614862	10009742.064362	none
level = 43, T = 2968, d = 671, jitter = 0.0943 UI rms, 0.4314 UI p-p_{max}				

Table 4.2 Terms and definitions used in pseudocode (Int=integer part, frac=fractional part)

TERM	DEFINITION	CONSTANT OR EQUATION
numel	number of elements in a frame	8
oh	general dedicated overhead bits	40
submax	maximum subframe size (bits)	512
T_{max}	maximum frame size (bits)	submax*8+8 = 4104
elmax	maximum element size (bits)	$\frac{T_{max} - oh}{numel} = 508$
f_t	tributary rate (f_t)	input variable
f_p	carrier payload rate (f_p)	input variable
Δ_t	tributary frequency tolerance in ppm	input variable
Δ_s	synchronous carrier frequency tolerance in ppm	input variable
$f_t$min	minimum effective tributary rate based on Δ_t and Δ_s	$f_t\left[1-\left(\Delta_t+\Delta_s*\frac{f_p}{f_t}\right)\right]$
$f_t$max	maximum effective tributary rate based on Δ_t and Δ_s	$f_t\left[1+\left(\Delta_t+\Delta_s*\frac{f_p}{f_t}\right)\right]$
level	level or feasible format number	incremented for each feasible format found
eltot	total bits in a data element	incremented each iteration
T	total bits per frame	oh+eltot*numel
f_f	UIPM frame rate	$\frac{f_p}{T}$
d	number of (fixed) data (D) bits per frame (8n-1)	$\text{Int}\left(\frac{f_t}{f_f}\right)$
n	number of data bits per data element	$\frac{d+1}{numel}$
fixed	number of fixed stuff (R) bits per frame (8k)	T-d-oh-1
ρ	nominal stuff ratio	$1-\text{frac}\left(\frac{f_t}{f_f}\right)$

4.3.2 Pseudocode of Format Design Algorithm

The following pseudocode describes the main kernel of the generic format algorithm. Table 4.2 provides a list of terms and definitions used in the pseudocode and the source code.

Procedure:

1. Input tributary rate (f_t), carrier payload rate (f_p), tributary and carrier tolerances ($\Delta_t$and Δ_s).
2. Input parameters for jitter simulation (phase threshold, PLL bandwidth, damping coefficient, and sample duration).
3. Initialize all variables, set level and eltot to 0.
4. **REPEAT**:
   i. Increment level count.
   ii. REPEAT:
   - increment eltot
   - T = oh+eltot*numel (find current total)
   - $f_f = \frac{f_p}{T}$ (find current frame rate)
   - $d = \mathrm{Int}\left(\frac{f_t}{f_f}\right)$ (define # of data bits based on f_f)
   - if $\mathrm{frac}\left(\frac{f_t}{f_f}\right) = 0$ AND $\mathrm{frac}\left(\frac{d}{\mathrm{numel}}\right) = 0$ then d = d-1

     This flags condition for ρ=0 and S=data while frac(f_t/f_f)=0 alone does not distinguish between ρ=0 and ρ=1.
   - $n = \frac{d+1}{\mathrm{numel}}$ {←"d+1" is due to (n-1)D in [S]} (define n based on # of data bits; n must then be an integral multiple of numel, i.e. frac(n)=0)

UNTIL either eltot = elmax {criterion #5} OR...

Criterion #1:

frac(n) = 0 AND... {← integral number of data bits in element}

Criterion #2:

eltot ≥ n AND... {← total bits per element ≥ data bits per element}

Criterion #3:

n ≥ 1 AND... {← there is at least one data bit per element}

Criterion #4:

$d*f_f \leq f_t min$ AND...

$(d+1)*f_f \geq f_t max$ {ensures adequate stuffing range}

iii. If design criteria is met then:

- fixed = T-d-oh-1
- $\rho_{nom} = 1 - frac\left(\frac{f_t}{f_f}\right)$, or if $frac\left(\frac{f_t}{f_f}\right) = 0$ AND $frac\left(\frac{d}{numel}\right) = 0$, then $\rho = 0$
- Simulate jitter for f_t from $f_t min$ to $f_t max$ in 10 Hz steps and find p-p_{max} value.
- Compare jitter of current format with previous format and save parameters of format with lowest p-p_{max} jitter.

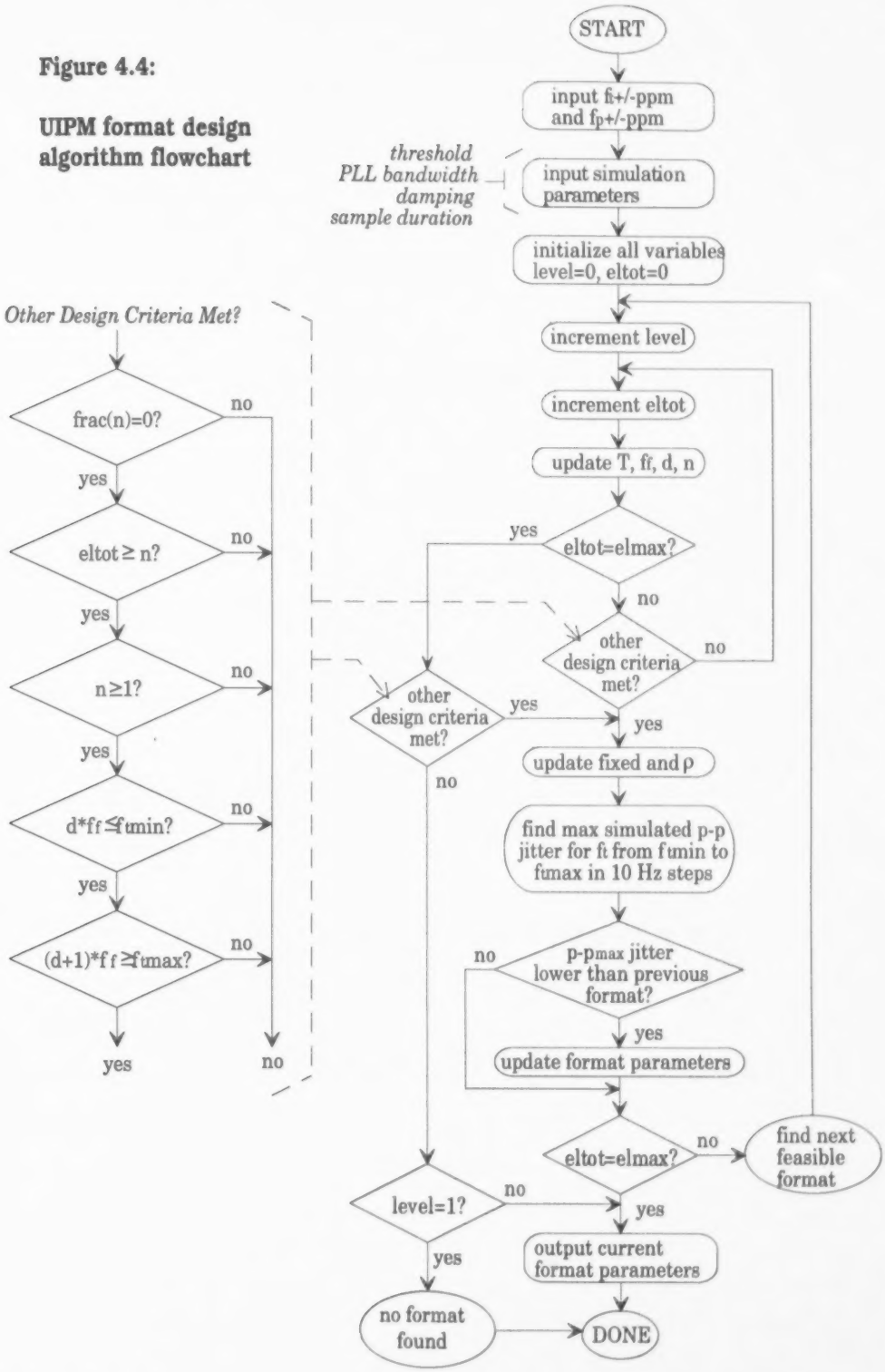
Else: design criteria was not met and no format was found for current level.

UNTIL eltot = elmax (design reaches maximum size)

5. Output: f_t, level, T, d, fixed, ρ, wideband p-p and rms jitter, and worst case narrowband p-p and rms jitter.

Figure 4.4:

UIPM format design algorithm flowchart



4.3.3 Flowchart of Format Design Algorithm

The flowchart shown in Figure 4.4 describes the rate adaptive mapping algorithm in a visual manner and provides a different perspective of the algorithm than given by the pseudocode. As in the pseudocode, the condition of eltot = elmax indicates the design has reached its maximum size. There are two iterative loops evident in Figure 4.4 indicated by arrows going back up the flow. The inner loop is the main iterative loop that determines a feasible format. The outer loop is a secondary loop that evaluates the jitter generated by feasible format designs.

4.4 WAITING TIME JITTER MODELING

4.4.1 General Description

The UIPM format design algorithm incorporates waiting time jitter simulation to select amongst feasible format alternatives that satisfy the format design criteria. The final selection from all feasible designs is based on minimum simulated jitter. Jitter simulation (as opposed to theoretical analysis) is used to assess jitter performance because of large amounts of fixed stuffing that is implicit in UIPM applications. It has been shown that in the presence of large amounts of overhead or fixed stuffing, the nominal stuff ratio is no longer an accurate predictor of waiting time jitter generation [GrMo86, GrMM87].

The jitter produced by a UIPM format is obtained by simulating the synchronizer/desynchronizer process utilizing pulse stuffing techniques at the transmit end to accommodate tributary and carrier tolerances, and a desynchronizer PLL circuit at the receive end for recovery of the original tributary signal. The simulation method, based on [MoKG91], uses time domain generation of a sample of the synchronizer waveform followed by convolution with the discrete-time impulse response of the desynchronizer PLL. (Appendix G includes examples of sampled synchronizer waveforms generated by

the jitter simulation process.) The PLL acts as a low pass filter in terms of phase, rejecting high frequency waiting time jitter components.

The jitter simulation algorithm calculates the phase accumulation between the tributary and carrier clocks once per frame of the UIPM format under investigation, then passes this phase sample to a discrete-time filter which models the desynchronizer PLL. This process continues until a transient duration has passed after which steady state samples exist at the filter output. The jitter arises from the pulse stuffing mechanism in the synchronizer which is modeled in the jitter simulation algorithm.

The end-of-frame phase calculation is given by the following expressions, where ϕ_i is the *i*th phase sample, $\Delta\phi=(f_p/f_t)-1$ is the relative change in phase during the carrier payload clock period when a time slot contains a data bit, and *thresh* is the phase threshold beyond which a stuff request is initiated:

If ϕ_i>thresh then $\phi_{i+1}=\phi_i+d*\Delta\phi$-fixed-oh-1

If $\phi_i\leq$thresh then $\phi_{i+1}=\phi_i+(d+1)*\Delta\phi$-fixed-oh.

These expressions are equivalent to those given in [MoKG91] which presents additional background and theory on jitter simulation.

The p-p and rms jitter values are obtained by performing statistical analysis on the steady state filtered jitter samples over an equivalent of 0.1 seconds of real time. The p-p value is simply the difference between the maximum and minimum filtered jitter samples while the rms value is obtained from the standard deviation of the accumulated filtered jitter samples. The frame-by-frame phase accumulation is normalized to the carrier payload clock period resulting in jitter values expressed in unit intervals (UI) instead of absolute time. The p-p and rms jitter values are scaled by the factor f_t/f_p to obtain jitter values in terms of tributary clock UI because it is the jitter on the recovered tributary clock that is of interest.

Similar statistical analysis is performed on the unfiltered jitter samples resulting in wideband p-p and rms jitter. Depending on the desynchronizer filter characteristics, the narrowband jitter is normally much less than the wideband jitter indicating the phase-smoothing operation of the desynchronizer PLL.

The jitter simulation algorithm exists as a utility routine within the main UIPM format design program and was also written in Pascal. Complete source code is given in Appendix B.

4.4.2 Flow Chart of Jitter Simulation Algorithm

Figure 4.5 summarizes the general description of the jitter simulation algorithm in the form of a flow chart. The flow chart shows two iterative loops containing similar procedures. The first loop is for the transient duration where the transient response of the filter is allowed to decay to a negligible level before steady state filter samples are accumulated in the second loop for 0.1 seconds of real time for statistical analysis.

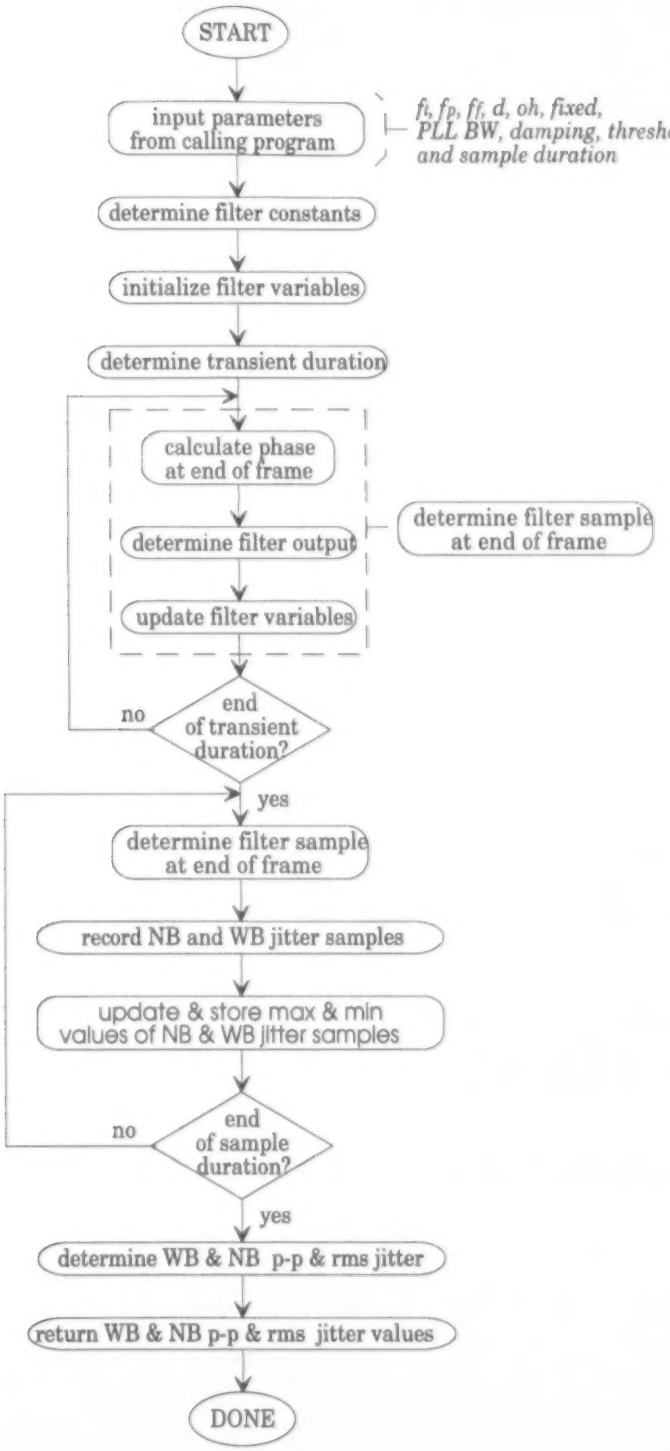

START
input parameters from calling program
fi, fp, ff, d, oh, fixed, PLL BW, damping, threshold, and sample duration
determine filter constants
initialize filter variables
determine transient duration
calculate phase at end of frame
determine filter output
update filter variables
determine filter sample at end of frame
end of transient duration?
no
yes
determine filter sample at end of frame
record NB and WB jitter samples
update & store max & min values of NB & WB jitter samples
end of sample duration?
no
yes
determine WB & NB p-p & rms jitter
return WB & NB p-p & rms jitter values
DONE

Figure 4.5 Flow chart for jitter simulation algorithm

4.4.3 Desynchronizer PLL Model

In implementing the jitter simulation algorithm, the phase response of the desynchronizer PLL was modeled as a second order low pass filter. Two versions of the jitter simulation algorithm were created, each with its own type of low pass filter. The transfer functions of these filters are given respectively by:

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{4-6}$$

and

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{4-7}$$

where Φout and Φin represent the output and input phase respectively, ω_n is the natural frequency in rads/sec, and ζ is the damping ratio. When ζ is $1/\sqrt{2}$, the transfer function in Equation 4-6 corresponds to a Butterworth filter [Best84]; for $\zeta \geq 1/\sqrt{2}$, no peaking occurs in the frequency response of H(s). Equation 4-6 represents an ideal PLL transfer function useful for jitter simulation of typical multiplex/demultiplex techniques, while Equation 4-7 models the transfer function of the actual PLL system constructed for obtaining experimental results.

For the above transfer functions the cutoff frequency (ω_c at -3dB from H(0)) is a function of the natural frequency and the damping coefficient. For the transfer function in Equation 4-6 the cutoff frequency is expressed as:

$$\omega_c = \omega_n \sqrt{1 - 2\zeta^2 + \sqrt{(2\zeta^2 - 1)^2 + 1}} \tag{4-8}$$

In this case, when $\zeta=1/\sqrt{2}$, $\omega_c=\omega_n$, while for $\zeta=1$, $\omega_c=0.6436\omega_n$. For Equation 4-7 the cutoff frequency is expressed as:

$$\omega_c = \omega_n \sqrt{1 + 2\zeta^2 + \sqrt{(2\zeta^2 + 1)^2 + 1}} \tag{4-9}$$

For this case, when $\zeta=1/\sqrt{2}$, $\omega_c=2.0582\omega_n$, while for $\zeta=1$, $\omega_c=2.4824\omega_n$. This relationship between the cutoff and natural frequencies is used in the UIPM algorithm to convert the desired cutoff frequency to the natural frequency which is passed as a

parameter to the jitter simulation algorithm. Derivations of equations 4-8 and 4-9 are given in Appendix C.

The transfer functions were implemented in the jitter simulation algorithm as Infinite Impulse Response (IIR) discrete-time filters of the following form using the bilinear transformation:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} - b_2 z^{-2}} \tag{4-10}$$

The bilinear transform maps the poles and zeros of H(s) in the s-plane into poles and zeros of H(z) in the z-plane and preserves the frequency response of the analog filter. The bilinear transform also ensures that a causal stable continuous-time filter maps into a causal stable discrete-time filter, except for quantization noise [OpSc89]. The discrete-time filter transfer function H(z) results in the following difference equation in the discrete-time domain that is easily realized in software:

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + b_1 y_{n-1} + b_2 y_{n-2} \tag{4-11}$$

The recursive relationship of Equation 4-11 is shown diagramatically in Figure 4.6 as a Direct Form I filter realization. The digital filter coefficients are functions of the frame rate (f_f) and are therefore scaled appropriately as the frame rate changes. A complete derivation of the analog to digital transformation and realization of the PLL transfer functions is given in Appendix D.

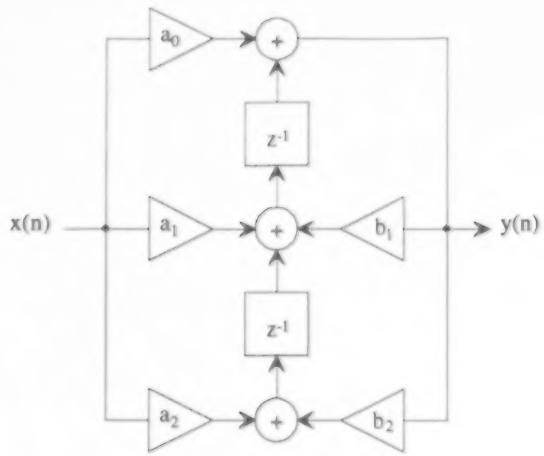


Figure 4.6 Direct Form I realization of IIR digital filter

4.4.4 Simulation Details

Pertinent details concerning the jitter simulation algorithm include the following:

- phase threshold between the tributary (write) and carrier (read) clocks that initiates a stuff request in the synchronizer;
- cutoff frequency and damping coefficient of the PLL transfer function;
- and transient and sample duration times for simulation.

For both versions of the jitter simulation algorithm all simulation details were identical except for the cutoff frequency and the form of H(s). The threshold was set at 1 UI (normalized to the period of the high frequency read clock). The cutoff frequency of the second order PLL transfer function was set at 350 Hz for the version implementing Equation 4-6, and at 1000 Hz for the version implementing Equation 4-7. Both versions used a damping coefficient of 1. This resulted in a critically damped system with a natural frequency of about 544 Hz for Equation 4-6 and about 403 Hz for Equation 4-7. The transient duration was set at 20 time constants allowing transients to die out almost completely and the sample duration was set at 0.1 seconds. This value for the sample duration gave almost identical jitter results compared to a trial run with the sample

duration set at 2 seconds but allowed much shorter run times. In addition, the 0.1 second sample duration provided an inherent high pass filter action with a cutoff frequency of 10 Hz thus excluding very low frequency jitter which is considered wander and which cannot normally be measured by laboratory equipment.

The time constant of the modeled second order PLL transfer functions is given as

$$\tau = \frac{f_f}{\zeta(2\pi f_n)}, \tag{4-12}$$

and hence the transient duration is dependent on the frame rate (f_f) of the format under investigation, instead of a fixed time. This is reasonable since the frame rate is also the absolute frequency of stuff opportunities, and of frame-end phase samples in the jitter simulation. This also ensured the transient duration would always be long enough but not longer than necessary.

In Equation 4-12, τ is the time constant, f_f is the frame rate (the reciprocal of the sampling time for the digital filter), and f_n is the natural frequency in Hz. For a continuous-time second order system described by equations 4-6 and 4-7 the time constant is defined as $\tau = \frac{1}{\zeta\omega_n}$, $0<\zeta<1$ [PhHa88]. For a discrete-time system described by Equation 4-10 the corresponding "digital" frequency is $\omega_n T_s$ where $T_s=1/f_f$ is the sampling period of the digital filter. This results in Equation 4-12 with the substitution $\omega_n = 2\pi f_n$. Another way of describing Equation 4-12 is that it converts the time constant in seconds of the continuous-time system H(s) into number of required samples at the frame rate for the discrete-time system H(z).

Equation 4-12 suggests that the number of samples during the transient duration should increase as the frame rate increases. Intuitively this makes sense because as the frame rate increases more samples are needed to span a specific unit of time (e.g. if f_f=100 Hz then 100 samples are needed to give one second of real time while for f_f=1kHz, 1000 samples are needed for the same time.)

Strictly speaking the expression for the time constant in Equation 4-12 only holds for $0<\zeta<1$, while the value chosen for the damping coefficient was $\zeta=1$. This expression, however, still provides an adequate method of relating the number of transient samples to the frame rate.

With the values for the cutoff frequencies and damping coefficient given above, Figures 4.7 and 4.8 show the frequency response of the PLL transfer functions H(s) described in equations 4-6 and 4-7 respectively. This represents the continuous-time characteristics of the PLL models used in the jitter simulation algorithm.

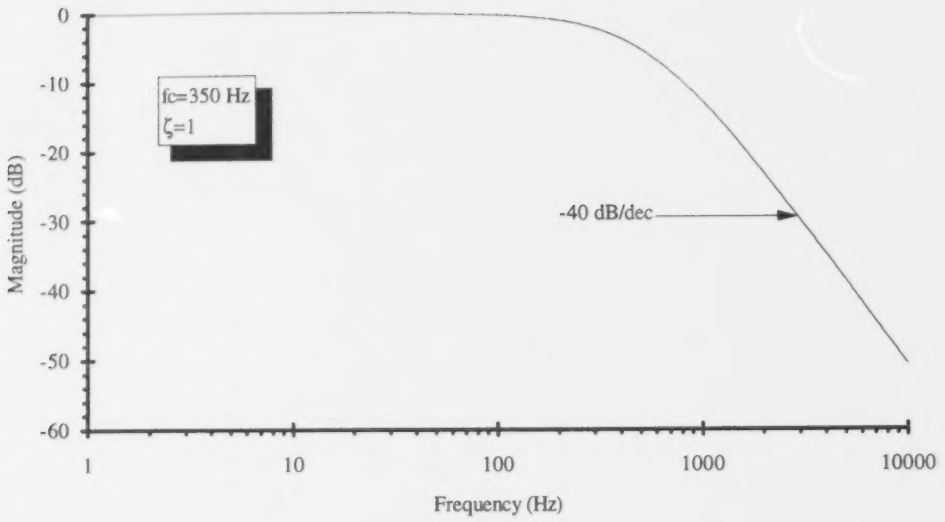


Figure 4.7 Frequency response of ideal desynchronizer PLL (Equation 4-6)

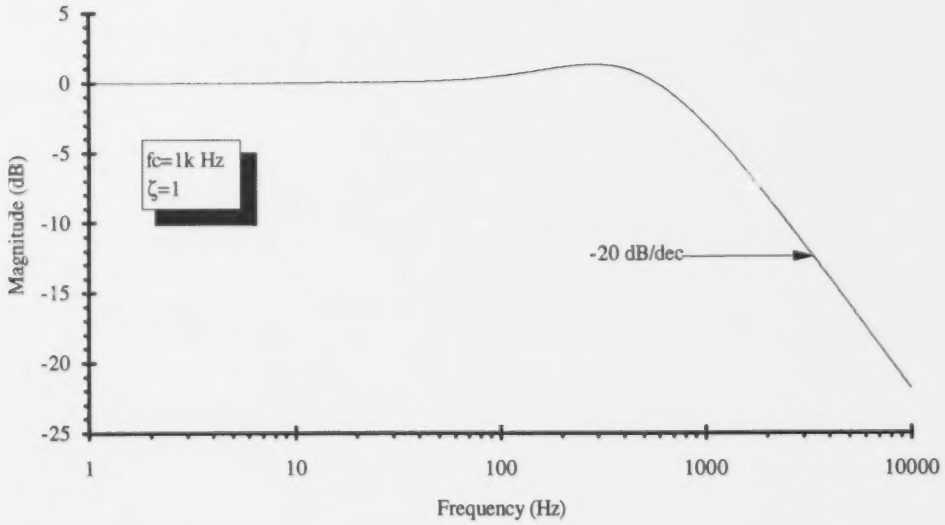


Figure 4.8 Frequency response of laboratory desynchronizer PLL (Equation 4-7)

4.5 VALIDATION OF JITTER MODELING

Validation of the jitter simulation algorithm was accomplished by comparing jitter simulation results with those given in [MoKG91] for the M12 multiplexer format, where 4 DS1 plesiochronous tributaries (1.544 Mb/s) are multiplexed to form a DS2 signal (6.312 Mb/s). In both simulation runs, predicted p-p and rms jitter was obtained over the specified DS1 frequency tolerance range of +/-130 ppm with a resolution of 1 Hz. Figure 4.9 shows the results of the jitter simulation algorithm described in Section 4.4 of this thesis and summarizes the simulation details. Figure 4.10 shows the results of the jitter simulation algorithm described in [MoKG91] (plot b) compared with a bit-by-bit sampling method (plot a) employed in [Say86].

There are two minor differences between the jitter simulation algorithm used in this thesis and the one in [MoKG91]. The one described in this thesis used the bilinear transform and accumulated 10,000 samples for each offset frequency. The algorithm in [MoKG91] used the impulse-invariant transform, which does not have the frequency preserving characteristics of the bilinear transform, and accumulated 4000 samples for each offset frequency except for the range from -2 to 12 Hz where 8000 samples were used and 10,000 samples at 7 Hz. Despite these differences, the two sets of results are virtually identical thereby verifying the jitter simulation modeling incorporated in the UIPM format design algorithm.

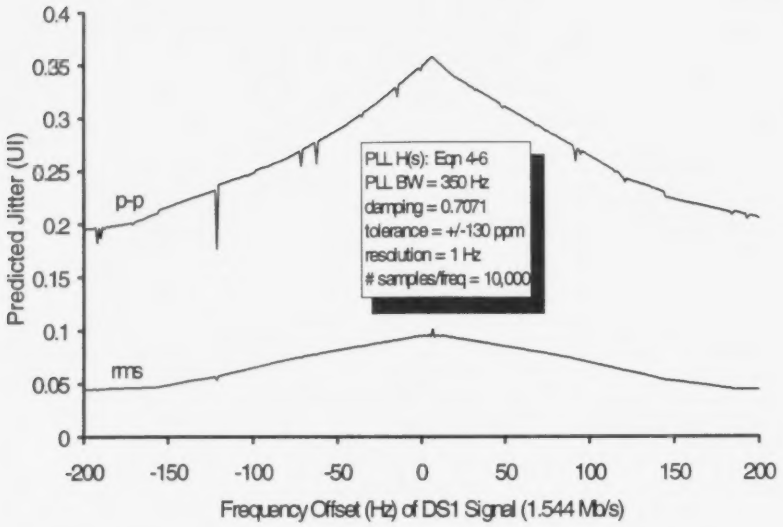


Figure 4.10 Predicted jitter versus DS1 offset frequency, using jitter model described in thesis

Figure 4.11 Predicted jitter versus DS1 offset frequency, from [MoKG91] a) bit-by-bit sampling, b) frame-synchronous sampling

4.6 SUMMARY

The UIPM format design algorithm determines a configuration of the generic format for a particular design case. In a complete UIPM system, parameters of the generic format generated by the algorithm would be passed to configurable UIPM multiplex hardware that performs the process of mapping a tributary signal into the payload of a standard carrier. The demultiplex hardware would use the same parameters to reverse the procedure.

There is a set of design criteria that must be met in the UIPM format design algorithm to determine all feasible format structures for a particular design case. This includes size constraints of the generic format, characteristics of the elements within the generic format, and accommodation of input and output frequency tolerances. A decision process based on minimum simulated jitter is used to select the best of all feasible formats as the optimum UIPM format.

The algorithm uses an iterative procedure to determine all feasible formats by incrementing the generic format size from minimum to maximum and simulating the jitter performance of those that meet the design criteria. Each feasible format corresponds to a consecutive level in the iteration process. Many, few, or possibly no levels, may be generated by a particular design case. The format design resulting in the lowest worst case simulated jitter is selected as the final generic format structure. Execution time is highly dependent on the number of feasible formats generated because the jitter simulation performed for each level requires the greatest amount of CPU time. Figure 4.2 summarizes the UIPM format design algorithm in the form of a flowchart.

The jitter simulation module uses time domain analysis involving frame-by-frame phase accumulation between the tributary and carrier clocks in the synchronization process of the multiplexer. The sequence of synchronizer phase samples are passed to a discrete-time filter that models the desynchronizer PLL in the demultiplexer. Steady state output

samples of the discrete-time filter are accumulated over an equivalent period of real time and used for statistical analysis resulting in p-p and rms jitter. A flowchart of the jitter simulation algorithm is given in Figure 4.3. Two separate versions of the jitter simulation algorithm were used, one for each of the two desynchronizer PLL models.

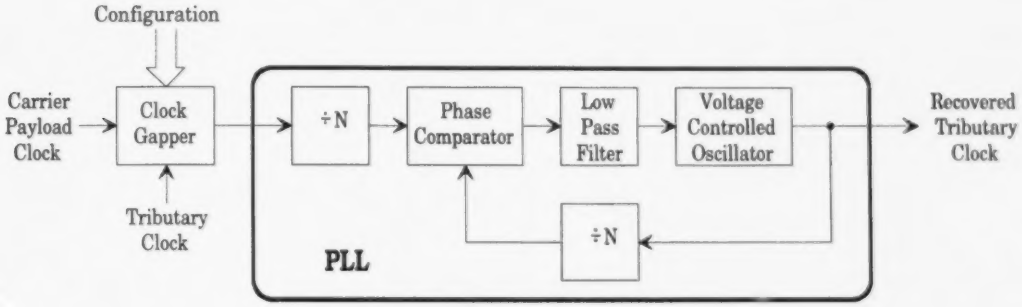
The synchronizer stuff request threshold of phase accumulation between the read and write clocks was set at 1 UI. The PLL was modeled as a discrete-time low pass filter with a cutoff frequency of 350 Hz for one version of the jitter simulation algorithm, and 1000 Hz for the other version. Both versions used a damping coefficient of one. Twenty time constants of the discrete-time filter were allowed to elapse before 0.1 seconds worth of samples were accumulated to determine the rms and p-p jitter values.

Verification of the jitter simulation model used in the UIPM format design algorithm was conducted by comparing predicted p-p and rms jitter results with those given in [MoKG91] for the M12 multiplex format. Virtually identical results verify that the jitter modeling developed in [MoKG91] was correctly implemented in the UIPM format design algorithm.

Chapter 5 concludes the detailed description of UIPM components developed in this thesis by discussing the experimental hardware used to obtain measured results and verify the UIPM concept. These results are compared with predicted results obtained from the UIPM format design algorithm and are presented in Chapter 6.

5 EXPERIMENTAL HARDWARE

This chapter describes the experimental hardware designed and constructed to verify the UIPM concept and to determine the feasibility of its implementation. The experimental hardware, shown in Figure 5.1, consists of a phase-locked loop (PLL) circuit and a clock gapping module that emulates relevant functions of the UIPM system.



Clock Gapper: Emulates functions in a UIPM muldem
PLL: Necessary for recovery of arbitrary-rate tributary clock

Figure 5.1 Experimental hardware block diagram

The key technical enabler for UIPM is the PLL because it provides the required jitter attenuation and is the most critical of all UIPM components. The clock gapping circuit, while fairly complex, is used here only to provide a clock input to the PLL that is gapped according to UIPM format characteristics. Hence, the nature of the laboratory experiments focus on the UIPM desynchronizer PLL.

There are significant differences between conventional desynchronization, used in typical multiplexer/demultiplexer (muldem) equipment, and UIPM desynchronization. Table 5.1 lists these differences and their consequences on PLL design.

Table 5.1 UIPM desynchronization versus conventional desynchronization

Conventional Desynchronization	Consequences	UIPM Desynchronization	Consequences
few, narrow gaps	• requires only basic PLL design • can use a narrow bandwidth LPF with constant filter parameters • PLL can be designed for specific jitter requirements	possibly very wide gaps	• requires addition of ÷Ns to PLL design • LPF parameters change as N changes • may encounter wide range of jitter levels
known tributary frequency	• can design PLL for specific rate and format • can use XVCOs	unknown tributary frequency	• must design PLL to cover a wide frequency range and avoid false harmonic lock • must use wide range VCO or switched banks of VCOs

The first section of this chapter describes the function and operation of the clock gapper which emulates multiplexing characteristics of UIPM formats. The next section discusses various aspects of the PLL system. This includes a description of the PLL as a whole, theoretical and practical design considerations, and descriptions of the individual hardware modules that make up the PLL circuit. Appendix E contains a complete set of schematics as well as a color plate of the experimental hardware.

5.1 CLOCK GAPPER

The function of the clock gapper is to gap or inhibit the rising edges of the carrier payload clock corresponding to both fixed and variable time slots in the UIPM signal that do not carry tributary data. Figure 5.2 portrays the principle of clock gapping.

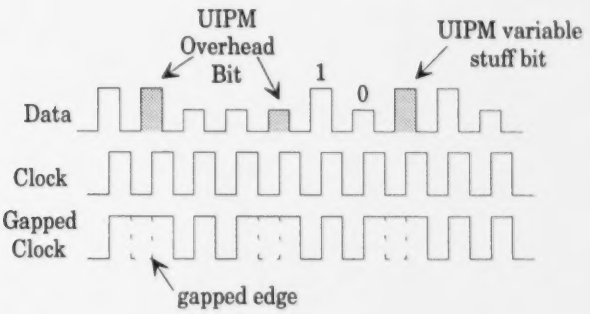


Figure 5.2 Clock gapping

In the laboratory system, the parameters determined by the UIPM format design algorithm are manually entered into the programmable logic in the clock gapper via dip switches. The clock gapping module can thus be configured to implement the time-base behavior of any particular UIPM format, as it is relevant to the desynchronization problem.

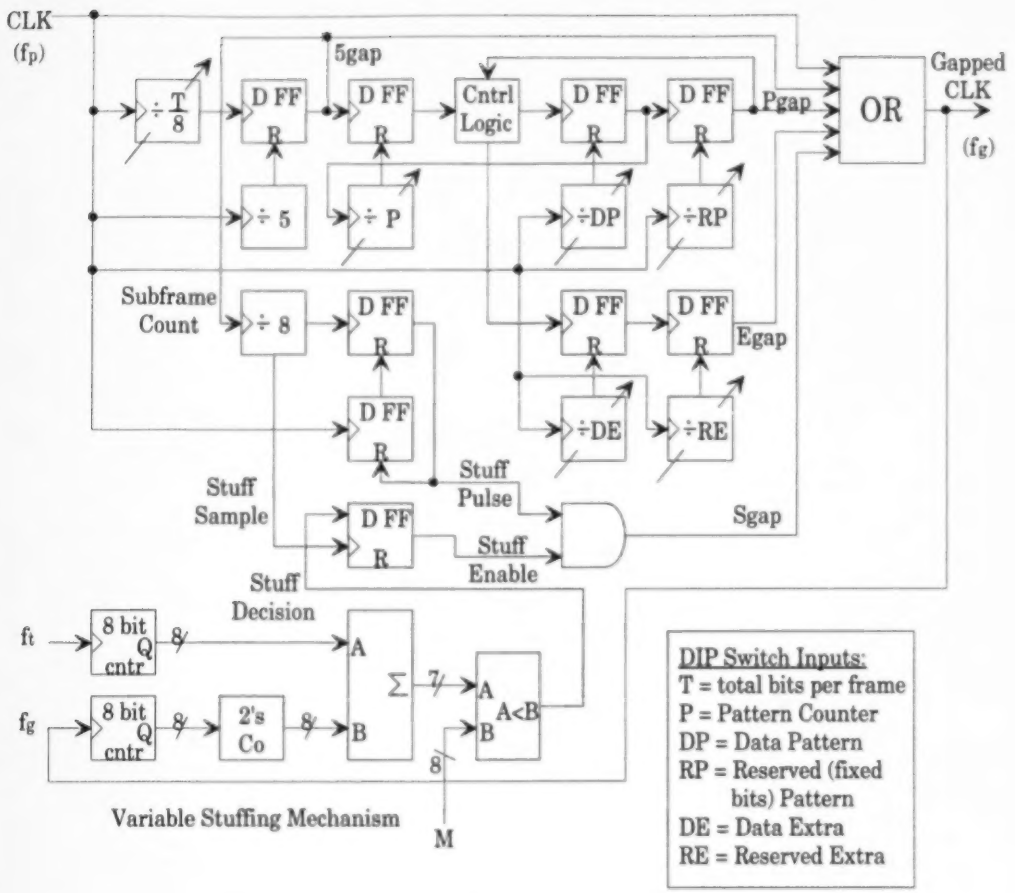


Figure 5.3 Clock gapper block diagram

Figure 5.3 portrays the clock gapper in detailed block diagram form. The clock gapping circuitry contains two major sections: a positive pulse stuffing justification mechanism, and programmable clock gapping logic. The outputs from these sections are OR'd together with the high speed carrier payload clock (f_p) to create the composite gapped payload clock (f_g). The following sections discuss the variable and fixed stuffing mechanisms on an individual basis.

5.1.1 Variable Stuffing Mechanism

The stuffing mechanism shown in Figure 5.3 emulates positive pulse stuffing justification by gapping the payload clock whenever the phase difference between the gapped payload clock (f_g) and the tributary clock (f_t) exceeds a set threshold. Instead of determining the continuous phase difference between f_g and f_t, this task is accomplished in a discrete fashion. Each clock signal is used to clock a respective 8 bit counter (an "f_t counter" and an "f_g counter"). The f_g count is 2's complemented and added to the f_t count to obtain their difference. The most significant bit of this result, which indicates the sign of the magnitude, is discarded to retain only the absolute difference allowing the counters to continually cycle through their count sequence from 0 to 255. Figure 5.4 illustrates this concept.

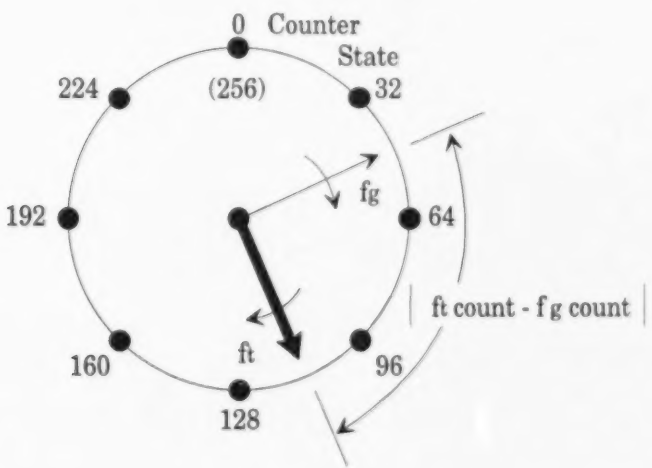


Figure 5.4 Phase comparison method used in variable stuffing mechanism

In Figure 5.4, the numbers around the circle indicate the state of the counter outputs. The thin pointer represents the current f_g count and the thick pointer indicates the current f_t count. Without the stuffing mechanism, $f_g > f_t$. With the stuffing mechanism operating, the gapped clock pointer will always lag behind the trib pointer by a set

amount. This is accomplished by comparing the absolute difference result with a binary number M in an 8 bit magnitude comparator. If the difference is less than M when the stuff decision is sampled then the payload clock edge corresponding to the next stuff opportunity is gapped.

5.1.2 Programmable UIPM Clock Gapping Logic

For purpose of gapping appropriate clock edges, the UIPM generic format can be thought of as eight identical subframes, ignoring the stuff opportunity in the last subframe which is controlled by the pulse stuffing mechanism. Therefore the majority of the clock gapping logic operates on a subframe basis. Figure 5.5 provides a simplified block diagram of the gapping logic.

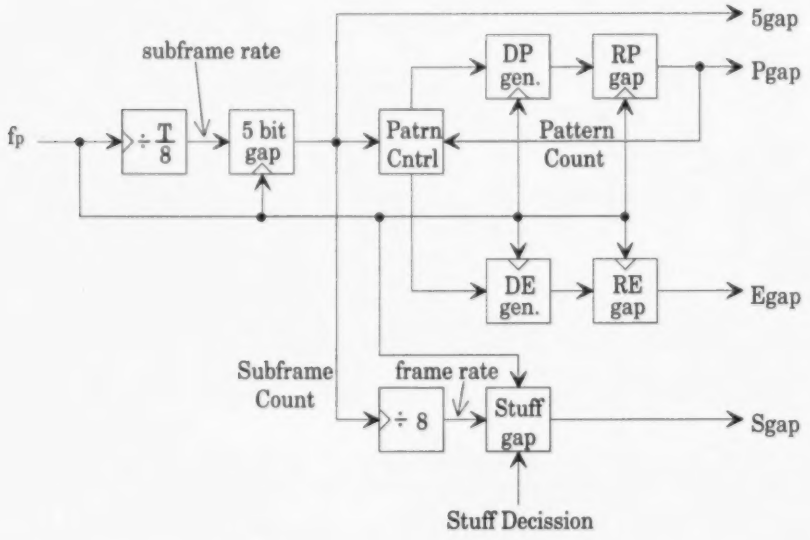


Figure 5.5 Simplified block diagram of UIPM clock gapping logic

The T/8 block generates subframe-rate pulses. This is a programmable divider where T is the total number of bits per frame determined by the UIPM format. The subframe pulses are used to trigger the fixed gapping sequence that occurs during each

subframe. The various gapping circuits in Figure 5.5 are typically composed of programmable divide-by-N circuits and D flip-flops as shown in Figure 5.3. Each divider is triggered by the output of the previous stage and stops after it reaches terminal count. The output of the associated D flip-flop remains asserted during the active period of the divider thus generating a gapping pulse as input to the OR gate.

For the experimental hardware, the information normally conveyed by the O_d and O_r bits of the UIPM generic format is determined by inspection and entered manually into the clock gapping programmable logic via dip switches. Each subframe (including the overhead bit preceding each subframe) is viewed as containing 5 dedicated overhead (OH) bits (plus one stuff opportunity, or S bit, in the last subframe), followed by a finite repeating pattern of data (D) bits and fixed stuff (R) bits, followed in turn by one extra set of D bits and R bits to complete the required number of D and R bits in each data element {nD, kR}. The clock gapping logic reflects this view of each subframe and allows the most distributed gapping sequence for a given n and k combination. For example, the sequence of bits in a subframe might be: 1OH, [4OH, {7D, 5R; 7D, 5R; 7D, 5R; 6D, 4R}], where [ ] indicates a subframe and { } indicates a data element. Here the pattern 7D, 5R repeats 3 times and the remaining D and R bits are accommodated by the extra set of 6D, 4R. This results in n=27 and k=19.

The distribution of D and R bits described above is programmed into the P, DP, RP, DE, and RE blocks of Figures 5.2 and 5.4. These are all manually programmable dividers where: DP and RP indicate the number of D and R bits in the repetitive pattern; P indicates the number of times the pattern is repeated; and DE and RE indicate the number of extra D and R bits required. The integer variables n and k discussed in chapters 3 and 4 are related to the hardware parameters in that n=DP+DE and k=RP+RE. Using this notation, for the above example: P=3, DP=7, RP=5, DE=6, and RE=4 (resulting in T=408).

Since all non-data bits generate a gap in the payload clock, the gapping sequence discussed above repeats at the subframe rate and gapping for pulse stuffing occurs at the frame rate. The 5 bit gapping block in Figure 5.5 generates a 5 bit-wide gap (5gap), then triggers the pattern control circuit into operation. The pattern control circuit triggers the DP generator to count DP clock cycles before triggering the RP gapping block to generate an RP-wide gap pulse (Pgap). The trailing edge of this pulse retriggers the DP generator and this "leap-frog" process continues until the pattern control counter reaches P counts upon which DP and RP blocks remain idle and the DE generator is triggered into operation. After DE clock cycles the RE block generates an RE-wide gap pulse (Egap). The entire system then waits for another subframe pulse to restart the process over again.

The divide-by-8 block generates frame rate pulses for the pulse stuffing logic. The leading edge of these pulses is coincident with the S bit in the UIPM format and triggers a circuit that generates a pulse one clock cycle wide. The trailing edge of the frame rate pulses is used to sample the output of the stuffing mechanism and generate a stuff enable (see Figure 5.3) for the next frame if the stuff decision output is asserted. If the stuff enable is true then the one clock cycle wide pulse is gated through to provide a stuff gap pulse (Sgap) coincident with a stuff opportunity. This provides frame-synchronous stuff threshold crossing detection that corresponds to the model of the synchronizer used in the jitter simulation algorithm [Moor88].

5.1.3 Verification of Clock Gapping Circuitry

The correct operation of the clock gapper was verified by using a frequency counter averaging over a 10 seconds period to measure the average frequency of the gapped clock output. A test tributary clock (f_t) was set at 10 MHz and the carrier payload clock (f_p) at the DS3 payload rate of 44.209694 MHz. The clock gapper was configured to implement the optimum UIPM format generated for this case which resulted in n=3 and k=5. The dip switches of the dividers in the clock gapper were set as follows: T/8=13,

P=1, DP=3, RP=5, DE=0, and RE=0.

Table 5.2 shows measured versus theoretical results for f_t = 10 MHz (ρ=nominal) as well as for $f_t$max (ρ=0) and $f_t$min (ρ=1) representing the extremes of the tributary frequency range the UIPM format can accommodate. The theoretical values for $f_t$max and $f_t$min were calculated using equations F-2 and F-3 from Appendix F and rounded up to the nearest Hz. These results confirm that the clock gapping module was operating in the desired manner.

Table 5.2 Measured versus theoretical results verifying correct operation of the clock gapper

Stuff Ratio (ρ)	Theoretical (MHz)	Measured (MHz, +/-1 Hz)
0 (variable stuffing off)	10.202237	10.202237
nominal	10.000000	10.000000
1	9.777144	9.777144

5.2 UIPM DESYNCHRONIZER PLL

The function of the desynchronizer PLL is to recover the arbitrary-rate tributary clock from the gapped payload clock (f_g) resulting in a smoothed read clock for the elastic store in the desynchronizer. The recovered clock must be phase synchronous to the original tributary clock, with phase jitter low enough to avoid overflow or underflow in the elastic store due to its finite size. In addition, the PLL must lock to inputs whose average frequency is some nominal value between approximately 1 MHz and 45 MHz while simultaneously avoid harmonic lock-in over this range.

The range of 1 MHz to 45 MHz for the UIPM desynchronizer PLL was deemed a reasonable a design objective for experimentation and proof-of-performance using a DS3 as the output carrier. The DS3 carrier payload rate of 44.209694 Mb/s results in a

theoretical range of tributary rates from 75.414 kb/s to 43.774422 Mb/s (see Appendix F) that can be carried by the UIPM generic format. A lower bound of 1 MHz was arbitrarily chosen to allow a practical limit on the PLL design.

The variable stuffing mechanism in the clock gapper ensures the average frequency of the gapped payload clock is equal to the frequency of the original tributary clock. The PLL generates a square wave clock at the same average frequency as the gapped clock and smoothes out the phase deviations caused by waiting time jitter. The clock output of the PLL is phase synchronous to the original tributary clock except for a small amount of residual jitter. Figure 5.6 illustrates this process.

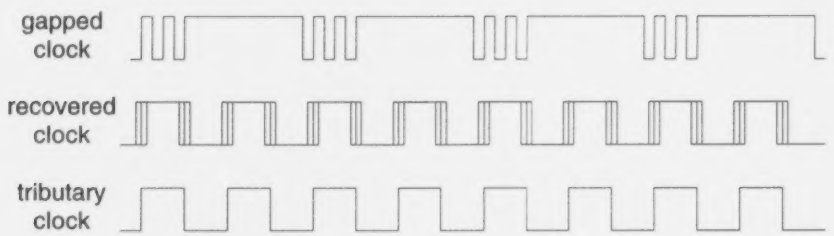


Figure 5.6 Waveforms in desynchronizer PLL

There are two distinct features of the desynchronizer PLL that allow it to derive the average frequency of the gapped carrier payload clock: a relatively narrow PLL bandwidth, and frequency dividers on both the input and feedback paths (see Figure 5.1). The PLL bandwidth (BW) is equivalent to the cutoff frequency (f_c in Hz) which is defined as the half power point (-3dB) of the PLL transfer function as discussed in Section 4.4.3.

A wide PLL bandwidth causes the output to track the input and provides little jitter attenuation. A narrow PLL bandwidth, however, creates a flywheel effect that retains clock information in the absence of an input and attenuates high frequency jitter (this feature is also common to regeneration type clock recovery circuits).

The distinguishing feature of the PLL used in the experimental hardware for UIPM is the need for the dual divide-by-N (÷N) circuits (Figure 5.1). This arrangement is typical of PLL circuits used as frequency synthesizers where each divider acts independently. In the desynchronizer PLL, however, the two dividers always divide by the same amount. The ÷N for the gapped clock ensures that periodic significant clock edges are presented to the phase detector in the PLL circuit, rather than the non periodic edges that exist in the gapped clock (an edge sensitive phase-frequency detector is used). In essence, the non periodic edges of the gapped clock are converted to periodic edges of a lower rate. The ÷N for the VCO output ensures the VCO frequency is correct and not a multiple or submultiple of the original tributary rate. Phase comparison can only be accomplished when the clock edges of the two signals being compared occur at the same rate. This action is illustrated in Figure 5.7.

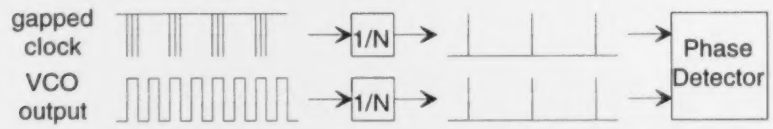


Figure 5.7 Use of dual ÷N blocks for proper phase detection

5.2.1 PLL Design

Figure 5.8 shows a block diagram of the PLL circuit used in the UIPM experimental hardware. It consists of three main sections constructed as separate units: an Analog Block, a Voltage Controlled Oscillator (VCO) Block, and a Divider Block. A description of each block is presented in Section 5.2.2.

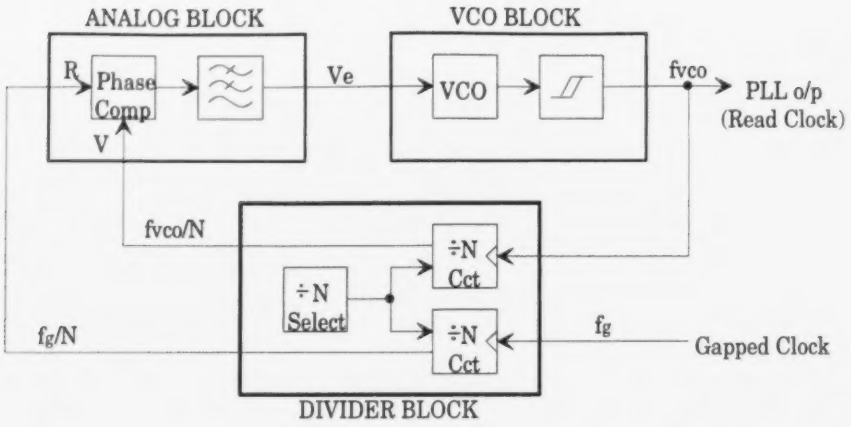
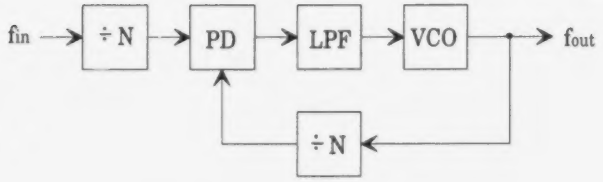
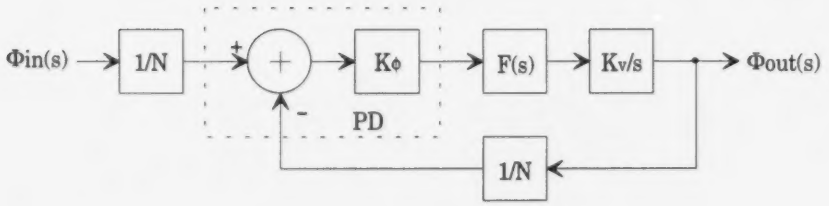


Figure 5.8 PLL block diagram



a) Functional block diagram



b) Linear model of the PLL

Figure 5.9 Simplified description of the PLL

To understand the operation and design of the PLL circuit, Figure 5.9 provides a simplified block diagram and linear model of the PLL. Referring to Figure 5.9a, both f_{in} and f_{out} are divided by the same amount before being applied to the phase detector (PD). This ensures that significant edges of f_{in} and f_{out} occur at the same rate. The PD is an

MC4044 which is an edge sensitive phase-frequency detector. Its output consists of periodic rectangular pulses (pump-up and pump-down signals) whose duty cycle is a function of the phase difference between the two inputs. The low pass filter (LPF) passes the low frequency content of the PD output to the VCO. The internal charge pump present in the 4044 is not used because greater speed can be achieved by using an external active filter [Moto89]. The LPF also determines the dynamic characteristics of the loop. The output of the LPF is an error voltage that is linearly related to the phase difference between the inputs of the PD. This error voltage is used to correct the VCO so that the frequencies of the inputs to the PD are equal with only a very small static phase difference existing between them.

Figure 5.9b provides a model of the PLL operating in its linear region. The functional blocks are replaced by Laplace Transform equivalents so that an expression for the transfer function can be obtained. The PLL can be analyzed like any feedback system except that the input and output are related by phase rather than amplitude. Thus the transfer function of the PLL is $H(s) = \frac{\Phi out(s)}{\Phi in(s)}$.

The transfer function of the PD is its conversion gain K_ϕ with units of volts/radian (v/rad) and the LPF has a transfer function F(s). The conversion gain of the VCO is K_v with units of rad/s/v. Since the radian frequency is the time derivative of phase, the phase difference between the inputs to the PD is proportional to the integral of error voltage applied to the input of the VCO. Therefore the transfer function of the VCO is K_v/s because it converts voltage to frequency and acts as an integrator for phase. In essence, the PD converts phase to voltage which is filtered by the LPF and then converted back into phase by the VCO as input to the PD after division by N.

The system equation of the PLL model can be written as [Best84]:

$$\frac{\Phi out(s)}{\Phi in(s)} = \frac{K_\phi K_v F(s)}{sN + K_\phi K_v F(s)} \tag{5-1}$$

The actual LPF used in the PLL is an active filter; its transfer function is:

$$F(s) = \frac{1 + \tau_2 s}{\tau_1 s} \tag{5-2}$$

where τ_1 and τ_2 are time constants of the RC networks in the filter circuit. Substitution of Equation 5-2 into 5-1 results in the overall PLL transfer function H(s):

$$H(s) = \frac{\frac{K\tau_2}{N\tau_1}s + \frac{K}{N\tau_1}}{s^2 + \frac{K\tau_2}{N\tau_1}s + \frac{K}{N\tau_1}}, \quad \text{where } K = K_\phi K_v \tag{5-3}$$

Comparing equations 5-3 and 4-7 (general transfer function) results in

$$\omega_n = \sqrt{\frac{K}{N\tau_1}} \tag{5-4}$$

and

$$\zeta = \frac{\omega_n \tau_2}{2}. \tag{5-5}$$

Solving for τ_1 and τ_2 results in PLL design equations

$$\tau_1 = \frac{K}{N\omega_n^2} \tag{5-6}$$

and

$$\tau_2 = \frac{2\zeta}{\omega_n}. \tag{5-7}$$

To determine τ_1 and τ_2, which in turn determine the values of the resistors and capacitors in the filter circuit, the values of K_ϕ, K_v, ω_n, ζ, and N must be known. K_ϕ and K_v are dependent on the phase detector and the VCO respectively and are known from measurements. As discussed in Chapter 4, ζ is set at 1, while ω_n is determined from Equation 4-8 where the cutoff frequency is set at 1 kHz (so ω_n=2531 r/s and τ_2=0.79 ms). The only variable that does not remain fixed is N which is determined from the output of

the UIPM format design algorithm.

5.2.2 Practical Design Considerations

There are two main design considerations that involve conflicting design requirements: the PLL BW and the value for N in the programmable dividers.

The PLL BW determines the extent of jitter attenuation. Ideally this should be as narrow as possible for maximum jitter attenuation. However, the PLL BW must be greater than the bandwidth of the VCO's frequency spectrum due to intrinsic phase noise, to maintain control of the VCO's phase and frequency [Gard79]. In addition, a narrow PLL BW results in a small value for ω_n requiring large values for τ_1 and τ_2 which may exceed practical design limitations for the active LPF. An investigation of the phase noise characteristics of the VCO circuits used for the experimental hardware led to the choice of 1 kHz for the PLL BW.

The value of N determines the time between phase comparisons in the PLL. Ideally, a large value of N would provide the best phase averaging because it would average out the effects of many fixed and variable stuff bits over a period of time. However, a large value of N reduces the frequency of phase comparison updates from the PD. Each update results in a pulse whose width is equal in time to the phase difference of the PD's inputs. With a large N, the pulse rate may not be high enough for adequate attenuation by the LPF for low tributary rates. This is known as *reference feedthrough* and could result in significant ripple on the VCO control voltage causing excessive phase modulation. The reference frequency must be greater than $\omega_2=1/\tau_2=1266$ r/s (or 201 Hz) of the LPF for maximum attenuation (since $\tau_2=0.79$ ms for $f_c=1$ kHz).

After initial experimentation, a small value of N was favored since it provided adequate phase averaging and attenuated reference feedthrough. The value of N chosen for most cases was n+1 where n is the number of data bits in a data element as determined by the UIPM format design algorithm. For cases where n was very small, a larger value

was chosen that still ensured reference feedthrough attenuation.

5.2.3 Hardware Modules

This section provides descriptions of the various modules that make up the PLL system.

5.2.3.1 *Analog Block*

The analog block shown in Figure 5.8 consists of a phase detector (PD) and a low pass filter (LPF). The PD/LPF circuit provides the basic function of converting the phase difference between the f_{vco}/N and f_g/N inputs into an error voltage Ve used as input to the VCO. Figure 5.10 shows the measured DC voltage output of the LPF as a function of the phase difference between the f_{vco}/N and f_g/N inputs to PD. This is the transfer function of the PD/LPF circuit. Because the transfer function is essentially linear over the entire range, the conversion gain (K_ϕ) of the PD/LPF circuit is constant at 0.5 volts/radian.

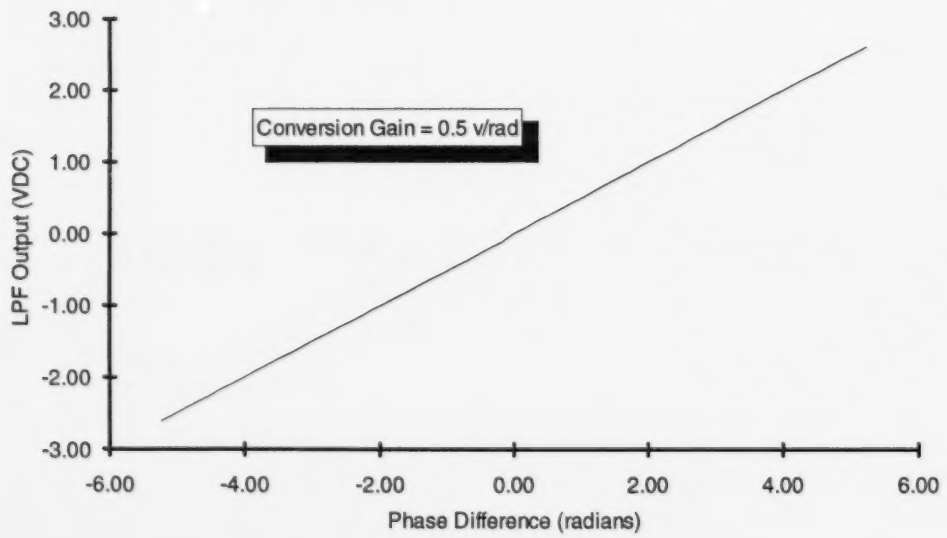


Figure 5.10 Transfer function of the PD/LPF circuit

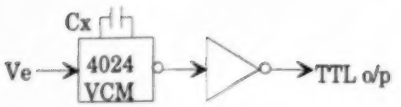
5.2.3.2 *Divider Block*

As shown in Figure 5.8, the divider block consists of two 16 bit divide-by-N counters whose parallel load inputs come from a common bank of dip switches used to set the value of N. One divide-by-N counter creates the f_g/N signal and the other one creates the f_{vco}/N signal.

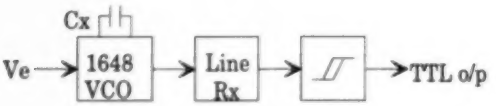
5.2.3.3 *VCO Block*

The VCO is the element under control in a PLL system. It converts the error voltage at its input into a frequency according to its conversion gain K_v. For the desynchronizer PLL the VCO is used to recreate the tributary clock by tracking the average frequency of the gapped clock.

One of the PLL design objectives was to provide recovered tributary clock rates between 1 MHz and 45 MHz. This requires the frequency range of the VCO to be the same resulting in a large tuning ratio of 45 to 1, but the tuning ratio of typical VCOs is about 3 to 1. In an attempt to avoid multiple VCO circuits or switched tuning components, a large-ratio VCO circuit based on a down-conversion method was designed and tested in the PLL system. It was found, however, that the intrinsic phase noise of this VCO circuit was significantly larger than the phase noise due to waiting time jitter itself (up to 8 times greater). Therefore, to obtain meaningful measurements of waiting time jitter, the large-ratio VCO module was replaced by two small-ratio VCO circuits shown in Figures 5.11a and 5.11b. Use of the large-ratio VCO for future research is considered in Chapter 7.



a) VCO block based on MC4024 VCM (1-10 MHz range)



b) VCO block based on MC1648 VCO (10-40 MHz range)

Figure 5.11 VCO block configurations

The two small-ratio VCOs exhibited very low intrinsic phase noise but covered limited ranges and required switching of their external tuning components to cover the desired range. The VCO circuit in Figure 5.11a was based on the MC4024 voltage controlled multivibrator (VCM) and covered the 1-10 MHz range while the one in Figure 5.11b was based on the MC1648 VCO and covered the 10-40 MHz range. The conversion gain of these VCO circuits depends on the value of the external tuning capacitor (Cx). Figures 5.12 and 5.13 show the transfer functions of the 4024 VCO and 1648 VCO respectively for the values of Cx used during experimentation. The dashed lines in these figures indicate the approximation used to determine the conversion gain (K_v) for each value of Cx.

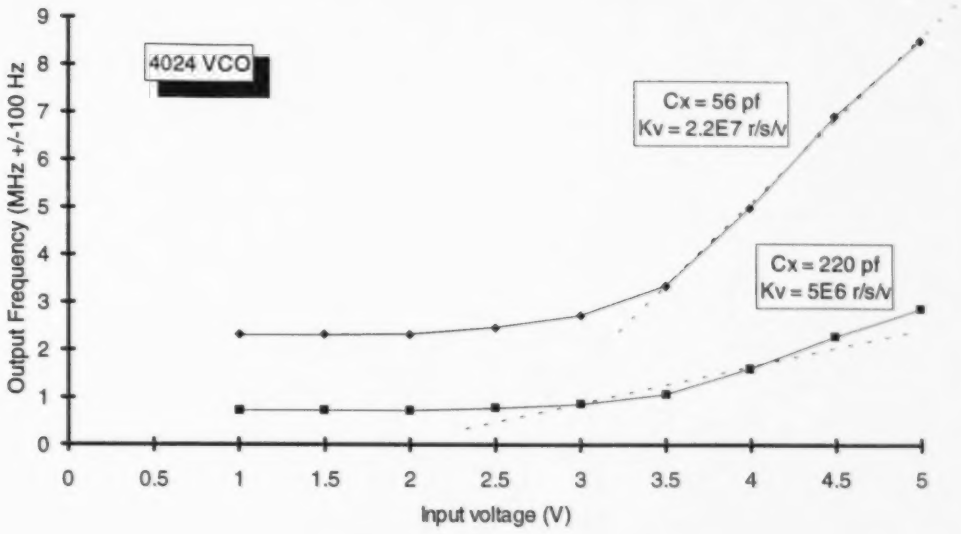


Figure 5.12 Transfer function of the MC4024 VCO for Cx=56pf and Cx=220pf

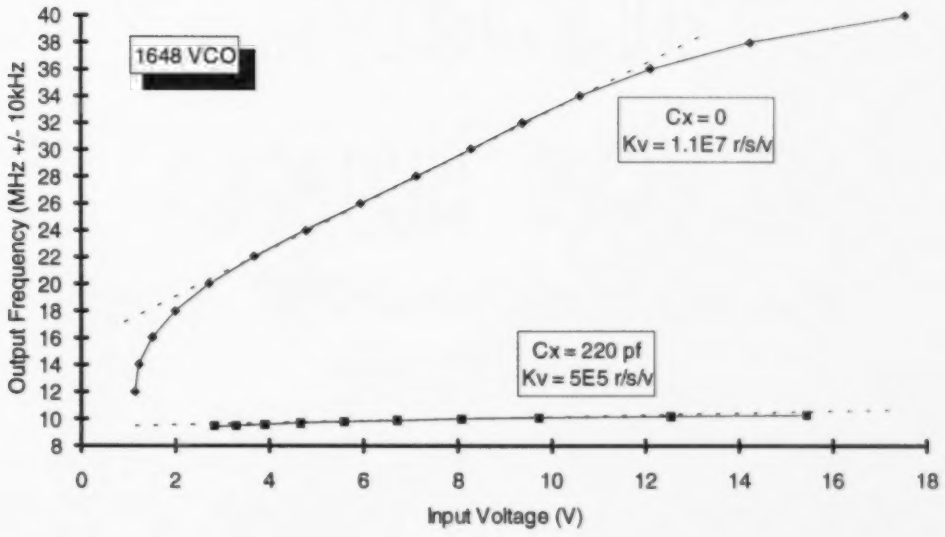


Figure 5.13 Transfer function of the MC1648 VCO for Cx=0 and Cx=220pf

5.3 HARDWARE PERFORMANCE

Figure 5.14 shows the equipment setup for testing the experimental hardware using available laboratory equipment. The HP8642A RF signal generator and HP3325A function generator are capable of less than 1 Hz resolution. The HP8642A provides a sinusoidal output as does the HP3325A for the 21-60 MHz range. For these cases the HP8116A function generators are used to convert the sine wave outputs into TTL outputs.

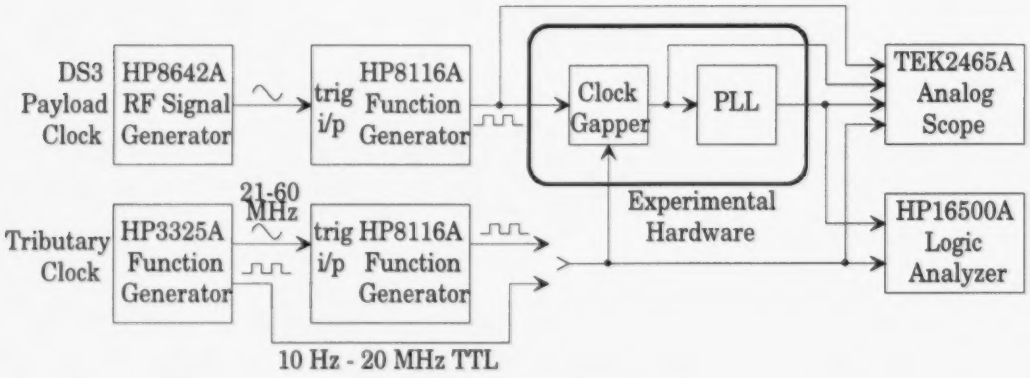
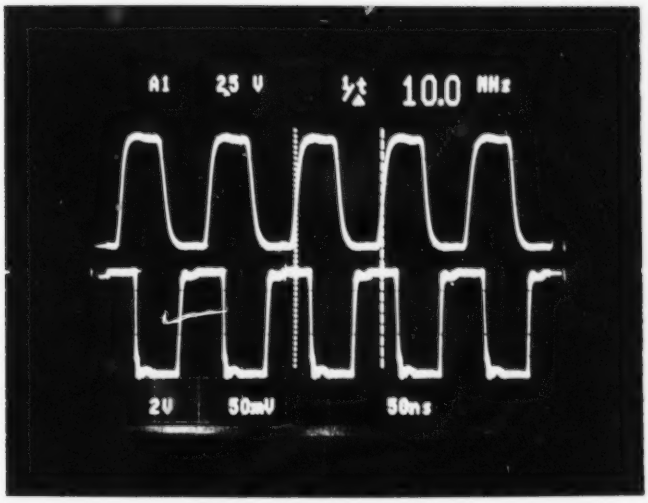
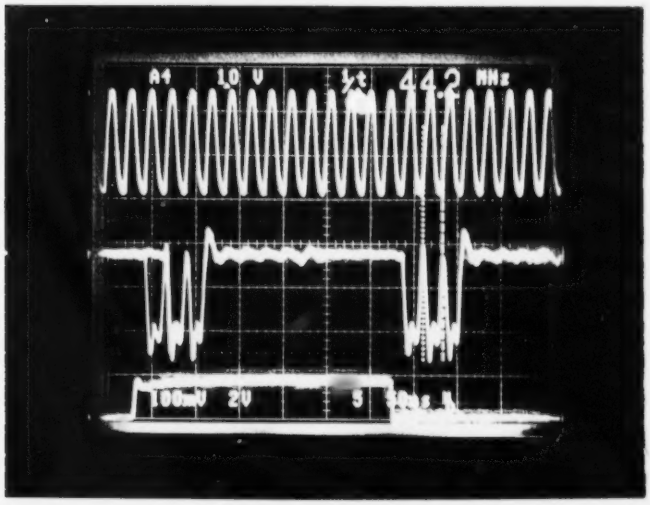


Figure 5.14 Equipment setup

Figures 5.15 and 5.16 show waveforms of the experimental hardware in operation. Waveforms in these figures were taken from an analog (non storage) oscilloscope and therefore do not accurately show the resulting jitter on the PLL output, but do give a relative indication of the hardware performance. In Figure 5.15a the top trace is an original tributary clock at 10 MHz and the bottom trace is the recovered (smoothed) tributary clock at the PLL (VCO) output. It can be seen that the PLL performs an adequate task of tributary clock recovery. It is important to stress that the 10 MHz recovered clock is derived from an intermediate gapped clock (due to UIPM transport) that has about 39% bulk gapping or, in other words, an equivalent signal with a 39% duty cycle where the "off" periods consist of clocking at 44.209694 MHz.

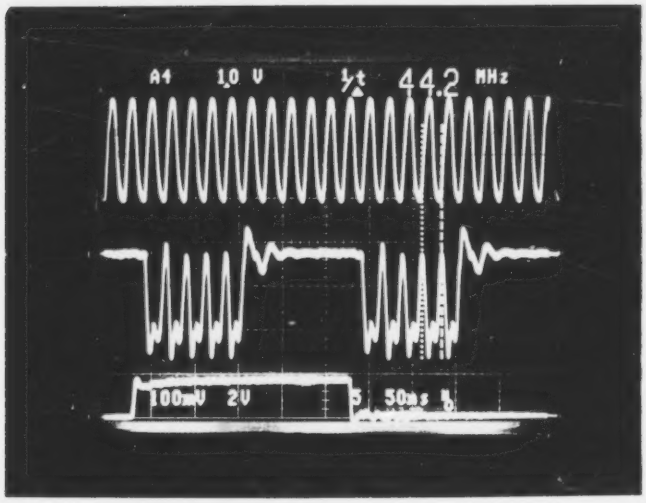


a) Original (top) and recovered (bottom) tributary clock

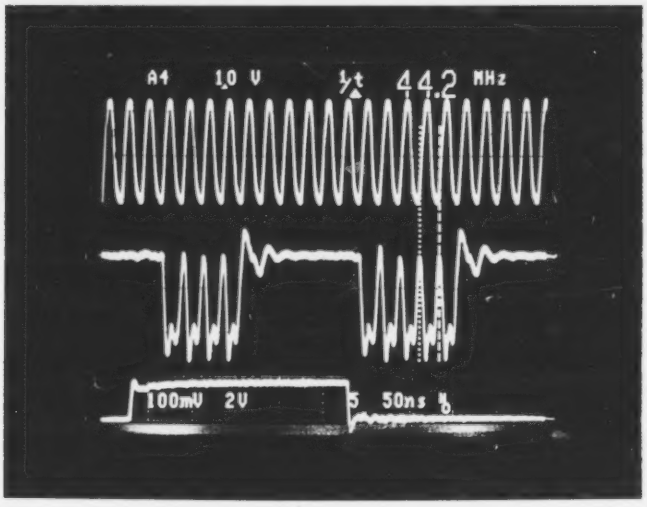


b) Original (top) and gapped (middle) carrier payload clock

Figure 5.15 Performance waveforms (f_t=10 MHz)



a) Original (top) and gapped (middle) payload clock for $\rho=0$



b) Original (top) and gapped (middle) payload clock for $\rho=1$

Figure 5.16 Performance waveforms (f_t=20 MHz)

Figure 5.15b shows the original (top) and gapped (middle) DS3 carrier payload clocks for the 10 MHz tributary case (the bottom trace shows the trigger signal) The distribution pattern of D and R bits for this case is: P=1, DP=3, RP=5, DE=0, RE=0. In other words, n=3 and k=5 in each data element {nD, kR}. The gapped clock waveform shows a stuffing subframe followed by a regular subframe. It is evident from this waveform that there are 3 rising clock edges, followed by 10 gapped clock edges (5 for the R bits and 5 for dedicated overhead bits).

The gapped clock waveform also shows the effect of the stuffing mechanism on the clock edge corresponding to the S bit - the very first rising clock edge of the trace. At times this clock edge is gapped and other times its is not. The percentage of time this clock edge is gapped is the numerical stuff ratio. The persistence of the oscilloscope's CRT causes many stuffing subframes to be displayed at once. Thus, a large stuff ratio would cause the first falling and rising clock edges to become faint while a small stuff ratio would cause the horizontal trace between the top of the first falling and rising edges to become faint. This is indeed observed as the 10 MHz tributary clock is varied slightly up and down in frequency.

Figure 5.16 shows illustrative payload clock and gapped clock waveforms for f_t=20 MHz. The gapped clock waveform in Figure 5.16a displays the case where the stuff ratio is 0 and the S bit's clock edge is never gapped (indicating the S bit is used for data). The gapped clock waveform in Figure 5.16b displays the opposite case where the stuff ratio is 1 and the S bit's clock edge is always gapped.

5.4 SUMMARY

Experimental hardware was designed and constructed to verify the UIPM concept and determine the feasibility of its implementation. The hardware consists of a clock gapper and a PLL. The clock gapper gaps the carrier payload clock in a manner

consistent with the UIPM format. This represents removal of all carrier payload bits that are not tributary data bits, including the variable stuff bits carrying dummy bits. The clock gapper consists of programmable gapping logic as well as pulse stuffing justification circuitry to accommodate the asynchronous relationship between the tributary and carrier payload rates.

The PLL recovers the tributary clock by locking to the average frequency of the gapped payload clock. The PLL is a second order system with a low pass transfer characteristic in terms of phase versus frequency. The PLL bandwidth is 1 kHz and the system is critically damped with a damping coefficient of 1.

The PLL system consists of an analog block, a VCO block, and a divider block. The analog block contains a PD/LPF circuit. The VCO block consists of two low noise VCOs, each covering part of the required operating range of the PLL.

Considerable effort was spent on development of the experimental hardware. Despite design and performance problems encountered during the development process, the experimental hardware performed well and provided meaningful results. Chapter 6 presents the measured results and compares them with predicted results obtained from the UIPM format design algorithm.

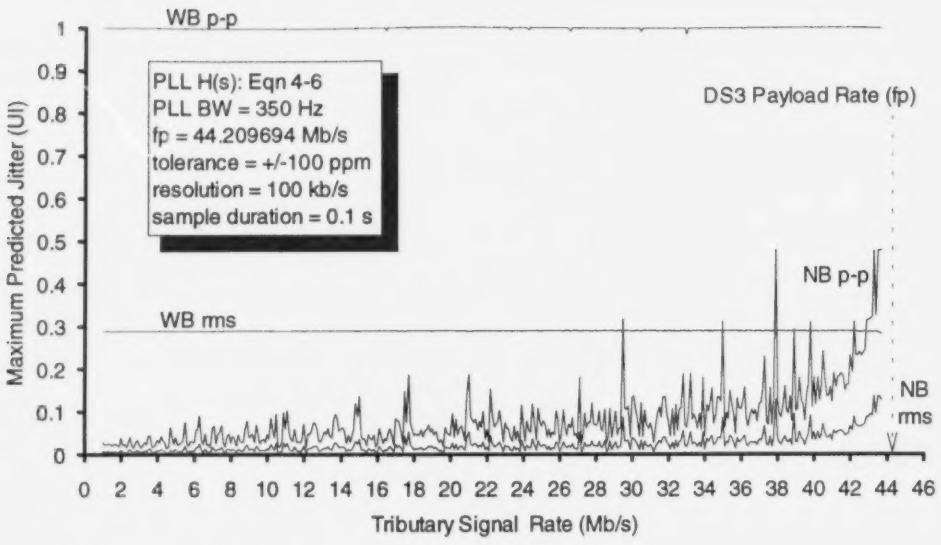
6 RESULTS

This chapter describes the predicted jitter performance of UIPM formats and measured results using the experimental hardware. An analysis of the predicted performance is given as well as a comparison between predicted and measured results.

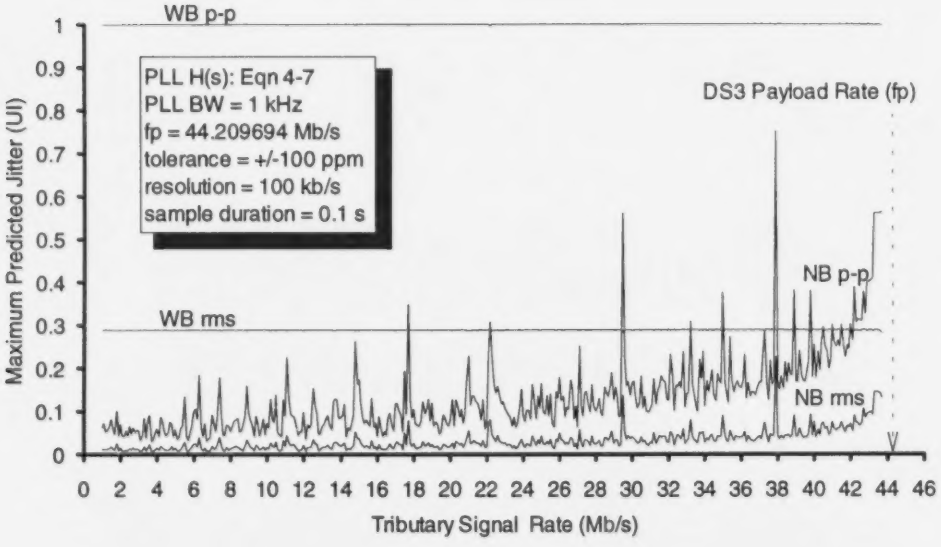
6.1 PREDICTED PERFORMANCE

The waiting time jitter simulated as part of the UIPM format design algorithm was captured for a range of UIPM designs to predict jitter performance. Figures 6.1a and 6.1b show the worst case predicted jitter performance for UIPM formats designed by the UIPM algorithm for a range of tributary rates adapted into the payload of a DS3 carrier. Both figures show the predicted wideband (WB) and maximum narrowband (NB) p-p and rms waiting time jitter for tributary rates starting at 1 Mb/s and increasing in increments of 100 kb/s to a maximum of 44 Mb/s. For each tributary rate, the jitter shown corresponds to the maximum NB p-p jitter occurring within +/-100 ppm of combined tributary and carrier offsets, in 10 Hz steps. Simulation details discussed in Section 4.4.4 are summarized in the shadowed boxes in these figures. Note that the predicted jitter is in units of UI normalized to the tributary clock period.

Figure 6.1a shows predicted jitter performance using a PLL model described by Equation 4-5 with a bandwidth of 350 Hz. This PLL model has a Butterworth type LPF characteristic with a bandwidth comparable to specifications in telecommunications industry standards [Bell86]. Figure 6.1b shows predicted jitter performance using a PLL model described by Equation 4-7 with a bandwidth of 1 kHz. This PLL model has a characteristic that closely resembles the actual PLL used in the experimental hardware. Its frequency response (Figure 4.6) has some gain near the cutoff point which accounts for special cases, discussed later, where the NB jitter actually exceeds the WB jitter.



a) using ideal PLL model with Butterworth type LPF characteristic



b) using model of PLL constructed for experimentation

Figure 6.1 Predicted UIPM jitter performance of optimum UIPM formats used to adapt tributaries into a DS3 payload

The discussion that follows will focus primarily on the predicted jitter performance depicted in Figure 6.1b since this relates directly to the experimental hardware. As shown in Figure 6.1b, the design algorithm was unable to generate formats to carry $f_t \geq 43.8$ MHz. For these frequencies the generic format structure reached its maximum size of 4104 bits before the format design criteria could be met. The largest theoretical tributary rate possible for the DS3 payload rate is 43.774422 Mb/s (and the smallest theoretical tributary rate is 75.414 kb/s - see Appendix F)

The largest p-p jitter value of 0.75 UI in Figure 6.1b was found at an input rate of 37.9 Mb/s (with 100 kb/s resolution). For this input rate the optimum UIPM format was only one increment (8 bits) smaller than the largest possible format size of 4104 bits. In this case a larger upper bound for the generic format size could result in an optimum UIPM format design with less jitter.

6.1.1 No-Format Rates

Predicted waiting time jitter was also captured for the same range of tributary rates as above, but in increments of 10 kb/s instead of 100 kb/s, giving a 10 times increase in resolution. This time there were a number of tributary rates within this range for which the UIPM format design algorithm was unable to generate any feasible formats. By coincidence, these "no-format" rates were not discovered with a 100 kb/s resolution. For the 10 kb/s resolution, the largest p-p jitter value of 1.02 UI was found at an input rate of 25.26 Mb/s. To further investigate the 25.26 Mb/s case, predicted jitter values were captured for tributary rates between 25.2 Mb/s and 25.3 Mb/s in increments of 1 kb/s. Figure 6.2 shows the results.

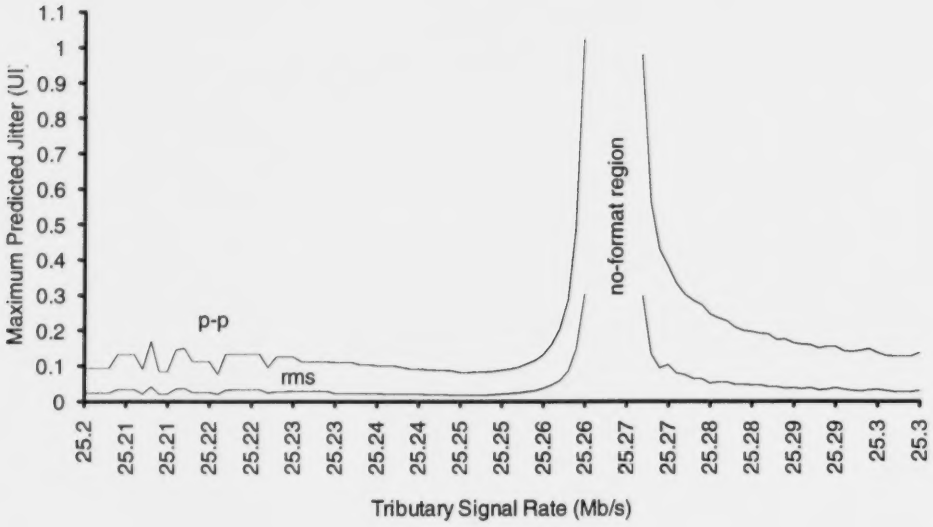


Figure 6.2 UIPM jitter performance of tributaries near a no-format region

In Figure 6.2 there is a no-format region where the UIPM format design algorithm was unable to generate any feasible UIPM formats for these rates. Just outside this region the predicted jitter is relatively high. To investigate the existence of no-format rates within the entire range of tributary rates (from 75.414 kb/s to 43.774422 Mb/s), all no-format rates were discovered using a resolution of 1 b/s. Two sets of no-format rates were generated, corresponding to two cases of f_p. In the first case, the payload rate was held constant at f_p=44.209694 Mb/s which is the rate when all 7 DS3 S bits are used for carrying data.

In the second case, the payload rate was made variable according to the following expression:

$$f_p = \frac{44.736 \text{ Mb / s}}{4760 \text{ bits}}(4697 \text{ bits} + u) \qquad (6\text{-}1)$$

where u = 0, 1, 2, 3, 4, 5, 6, or 7 DS3 variable S bits used for carrying data. In Equation 6-1, 44.736 Mb/s is the DS3 signal rate, 4760 is the total number of bits in a DS3 master

frame, and 4697 is the total number of dedicated data bits in a DS3 master frame. For determining all no-format rates for this case, the UIPM format design algorithm was modified so that a feasible format was sought among all 8 possible payload rates given by Equation 6-1 before a given tributary rate was declared a no-format rate. In other words, if a feasible format could not be found using no DS3 S bits, then the algorithm was run again using 1 DS3 S bit. If that did not produce any feasible formats, then 2 DS3 S bits were used and the algorithm was re-run. This continued until all 7 DS3 S bits were used. If no feasible formats were found for any number of DS3 S bits used, the tributary rate was declared a no-format rate. Although this modification to the UIPM format design algorithm is specific to the DS3 case, any plesiochronous multiplexing format containing one or more variable stuffing bits has the potential of adding similar flexibility to the UIPM generic format.

Table 6.1 shows the results for both cases of f_p. The table is divided into 3 major columns. The first major column lists the no-format tributary rates (in 1 b/s increments) for the payload rate case where all 7 DS3 S bits are utilized (fp=44.209694 Mb/s). The second major column lists the no-format tributary rates for the payload case where the number of DS3 S bits utilized vary from 0 to 7. Both of these major columns also list the number of discrete no-format tributary rates in each row of the table. The third major column indicates the approximate relationship of the no-format rates to f_p=44.209694 Mb/s. It is evident that the largest no-format region occurs for tributary rates close to 1/2 the payload rate. It is possible that the rates on either side of the x/y=1/2 no-format region exhibit the highest jitter of all rates in the design space but this was not evident when a resolution of 10 kb/s was used for capturing jitter.

Table 6.1 No-format rates

7 DS3 S bits used for data (f_p = 44.209694 Mb / s)		# DS3 S bits used = 0 to 7 (see Equation 6-1)		Approximate Fraction of f_p
No-format Rate (Mb/s) (1 b/s increments)	Number of rates	No-format Rate (Mb/s) (1 b/s increments)	Number of rates	
≤0.075413		≤0.075413		
5.525660-5.526764	1105	5.525660	1	1/8
6.535040-6.537850	2811	6.535040	1	1/7
7.367546-7.372638	5093	7.367546	1	1/6
8.841055-8.849337	8283	8.841055	1	1/5
11.051319-11.064344	13026	11.051319	1	1/4
12.630079-12.634161	4083	12.630079	1	2/7
14.735092-14.756029	20938	14.735092	1	1/3
16.776978-16.780293	3316	16.776978	1	3/8
17.682110-17.692122	10013	17.682110	1	2/5
18.945118-18.950443	5326	18.945118	1	3/7
22.102637-22.139378	36742	22.102637-22.106432	3796	1/2
25.260157-25.266760	6604	25.260157	1	4/7
26.523165-26.534933	11769	26.523165	1	3/5
27.628296-27.633822	5527	27.628296	1	5/8
29.470183-29.494102	23920	29.470183	1	2/3
31.575196-31.583065	7870	31.575196	1	5/7
33.153956-33.171360	17405	33.153965	1	3/4
35.364219-35.377781	13563	35.364219	1	4/5
36.837728-36.848701	10974	36.837728	1	5/6
37.890235-37.899355	9121	37.890235	1	6/7
38.679615-38.687351	7736	38.679615	1	7/8
≥43.774423		≥43.774423		
Total:	225225	**Total:**	3815	
% of total trib range:	0.527%	**% of total trib range:**	0.009%	

The number of tributary rates tested for the no-format condition totals 42,774,223. As Table 6.1 shows, the percentage of no-format rates to this total is 0.527% for the first payload rate case (all 7 DS3 S bits used) and only 0.009% for the second payload rate case (# of DS3 S bits varies from 0-7). This demonstrates that in either case, more than 99% of all tributary rates (at 1 b/s resolution) could be accommodated by the UIPM format design algorithm. The ability to easily modify the UIPM design algorithm to vary

its use of available S bits in a given standard format, combined with other technologies such as stuff threshold modulation [GrMM87], enhance the ability of UIPM to be truly *universal*.

No-format rates from 1 Mb/s to 50 Mb/s, in increments of 1 b/s, were also captured using an arbitrary carrier payload rate of 50 Mb/s to ensure that tributary rates precisely related to the carrier payload rate would be tested. This was done because it was not possible to test tributary rates having a precise fractional relationship to the DS3 payload rate. The results for the 50 Mb/s case confirmed the general finding with the DS3 payload rate case (first major column in Table 6.1) in terms of the relationship of no-format rates to carrier payload rate and the percentage of no-format rates in the design space.

An interesting observation is that all of the no-format rates recorded in Table 6.1 are related to f_p such that $f_t \approx f_p(x/y)$ where $x \in \{1,2,3,...,7\}$ and $y \in \{x+1,...,8\}$. The fraction x/y can be described more visually for all combinations in the following manner:

1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8
2/3, 2/4, 2/5, 2/6, 2/7, 2/8
3/4, 3/5, 3/6, 3/7, 3/8
4/5, 4/6, 4/7, 4/8
5/6, 5/7, 5/8
6/7, 6/8
7/8

Note that the denominator of the fraction x/y does not exceed 8. This is because there are 8 subframes per frame of any UIPM format. The fraction x/y means there are *x* tributary bits for every *y* carrier payload bits (consisting of the rest of the UIPM frame). Alternatively, x/y represents the ratio of data bits to total bits in one UIPM frame. Since there are 8 subframes in a UIPM frame, the total number of bits per frame will always be a multiple of 8. For the no-format rates, the UIPM format design algorithm will never be able to find a valid format size where the ratio of data bits to total bits results in x/y as given above *and* where the number of data bits per frame meets the first and fourth format

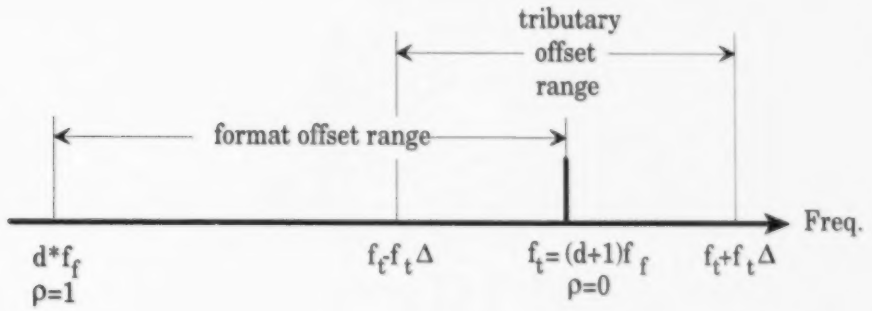
design criterion given in Chapter 4. These criteria state that the number of data bits per frame must have an integer part that, when 1 is added to it, is a multiple of 8, and that there must be a fractional amount which avoids a stuff ratio of 1 or 0. However, for $y \in \{9,10,11,...\}$, even though ratios of data bits to total bits exist that violate the first and fourth criterion (e.g. x/y=1/10=8/80), there are other format sizes that do meet all the criteria (e.g. x/y=1/10=7.2/72=15.2/152 etc.).

One possible way to avoid this problem is to use positive/zero/negative (+/0/-) stuffing where two stuff bits are used instead of one in a UIPM frame. In this method the stuff ratio varies between -1 and 1 and is 0 when the tributary rate is at its nominal value. The benefit is that the tributary offset range that the format can accommodate is increased. This approach is further discussed in Chapter 7.

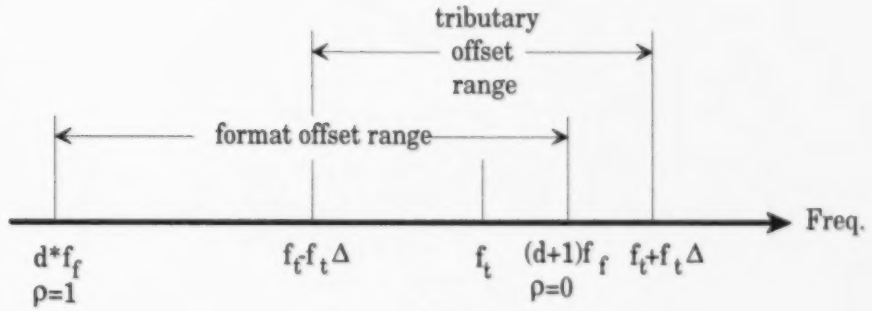
Initially it was thought that the 1 b/s resolution was not fine enough to allow no-format rates to be captured where $y \in \{9,10,11,...\}$. But additional testing for no-format rates with a resolution of 0.1 b/s did not produce any new results. However, capturing all no-format rates using a modified UIPM generic format structure consisting of 9 subframes instead of 8, keeping all other features the same, resulted in the addition of x/9 to the set of fractions represented by x/y. Therefore, the values of x and y can be stated more generally as: $x \in \{1,2,3,...,\#subframes-1\}$, $y \in \{x+1,...,\#subframes\}$.

For all no-format rates less than the maximum limit imposed by the UIPM generic format structure, the UIPM format design algorithm was, in general, unable to find any feasible formats because the frequency tolerance criterion (fourth) could not be satisfied. There are two cases where this can occur. One is where the variable stuff bit is needed for carrying data at the nominal rate ($\rho=0$ and $[d+1]f_f=f_t$) and no variable stuffing capacity is left to accommodate the tolerance. The other case is where the variable stuff bit is always stuffed with a dummy bit ($\rho=1$, $d*f_f=f_t$) to carry data at the nominal rate and again no variable stuffing capacity is left. This leads to the conclusion that there must be a totally non-harmonic relationship between f_p and f_t. However, even without such an explicit

relationship, this prohibits very few (if any) UIPM applications in practice. If necessary, stuff-threshold modulation schemes [GrMM87] or +/0/- stuffing may allow such circumstances to be accommodated.



a) $f_t=f_p(x/y)$



b) $f_t \approx f_p(x/y)$

Figure 6.3 Tributary and format offset ranges for no-format rates

Table 6.1 also shows that the no-format tributary rates occur in small ranges of rates that are approximately equal to a fraction of the payload rate given by x/y previously discussed. This is because either the maximum or minimum boundary of the tolerance range offered by the UIPM format for such cases lies within the tributaries tolerance range instead of outside. For tributary rates that are an exact no-format fraction of the payload rate ($f_t=xf_p/y$), one boundary of the format tolerance will be at the tributaries nominal rate

as shown in Figure 6.3a. For other no-format rates within a no-format region ($f_t \approx x f_p/y$), one of the boundaries of the format's offset range will be inside the tributary's offset range as shown in Figure 6.3b.

The format's offset range lies between $d*f_f$ and $(d+1)*f_f$ where d is the number of data bits in one frame of a UIPM format. Since d is the integer part of a real number, it will remain constant for a number of tributary rates that are very close. Thus a number of tributary rates close to a rate that is an exact no-format fraction of the payload rate will also be no-format rates for a given tributary offset range.

6.1.2 General Analysis and Observations

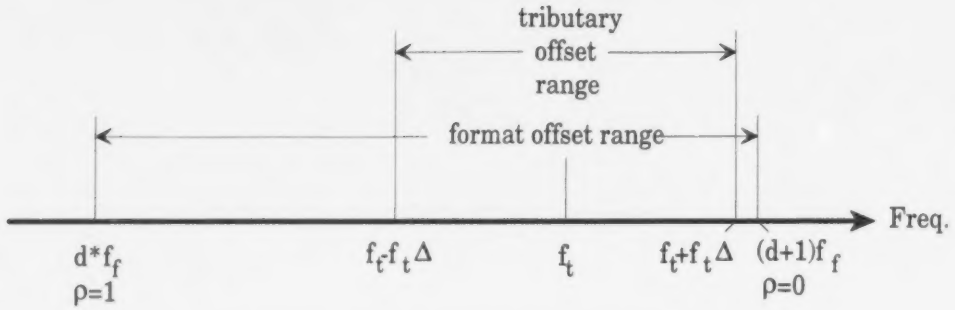
An interesting and unexpected set of results concerns the predicted WB p-p and rms jitter shown in Figures 6.1a and 6.1b. For all UIPM format designs, (indeed for any positive pulse stuffing based format such as the DS1 or DS3) the predicted WB p-p jitter is always about 1 UI and the predicted WB rms jitter is always about 0.2886 UI (tributary clock UI in both cases). The reasons for such consistency are not immediately obvious. Intuitively, a consistent WB p-p jitter of 1 UI seems sensible because it suggests the stuffing mechanism is operating in a manner that ensures the 1 UI p-p limit is never exceeded. However, reasons for the consistent value of 0.2866 UI rms for WB jitter, regardless of format structure, are less obvious.

Appendix F attempts to explain these results in a quantitative manner. To summarize Appendix F, the WB p-p jitter is always about 1 UI because no matter what the absolute p-p phase difference is between the read and write clocks in the synchronizer portion of a multiplexer, the conversion to tributary (read) clock periods always results in a theoretical value of 1 UI. The reason the WB rms jitter is always about 0.2886 UI is because the frame-by-frame phase waveform is always equivalent to a scaled triangular waveform with a p-p amplitude of 1. The standard deviation calculation in the jitter simulation algorithm used for determining the rms jitter removes the DC component of the

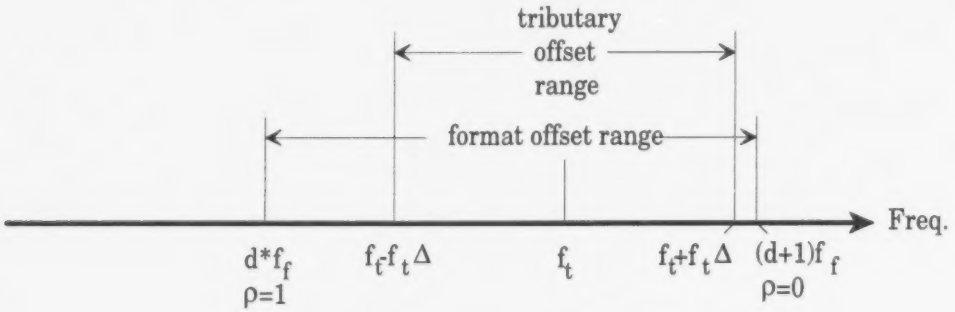
waveform and effectively computes the rms AC of the waveform which, for any unit amplitude triangular waveform, is always 0.288675.

Another interesting observation is that the trend in Figure 6.1 shows an increase in jitter as the input rate approaches the DS3 payload rate. This is because the optimum UIPM formats tend to be larger as f_t increases. As the frame size increases, the stuff opportunity rate (f_{so}) decreases, causing more low frequency jitter components to fall within the desynchronizer passband. The optimum format size increases as f_t approaches f_p because less fixed stuffing is required, necessitating larger format sizes to create adequate data capacity. In other words, larger formats are needed to maintain the proper ratio of data to non-data bits that results in the correct payload rate.

In any particular case, however, the optimum format may be larger than smaller feasible formats which offer higher f_{so}. This is because the nominal stuff ratios of the smaller formats are very close to 0 or 1 resulting in variable stuffing patterns that cause considerable low frequency jitter within the passband of the PLL. Figure 6.4a shows this case. The UIPM design algorithm is therefore forced into larger format sizes that actually reduce the format's offered tolerance range but result in a nominal stuff ratio that is not close to 0 or 1. This generates a variable stuffing pattern with less in-band jitter power. This case is shown in Figure 6.4b. The overall result is lower worst case jitter within the tributary's tolerance range even though f_{so} is lower and jitter at nominal rates may actually be higher. Formats larger than the optimum always exhibit more jitter because the stuff opportunity rate decreases further and because the boundaries of the tributary's and format's offset ranges become too close causing higher jitter at the offset extremes.



a) a small format can have high jitter if the tributary's offset range is too close to a ρ=0 condition



b) a larger format will have less jitter because the nominal stuff ratio is not close to 0 or 1

Figure 6.4 Case where a large format can have less jitter than a small one

To observe the action of the UIPM format design algorithm, Figures 6.5 to 6.17 show the maximum predicted p-p waiting time jitter (narrowband) versus format size for various selected tributary rates. This represents the intermediate worst case jitter predictions the algorithm considers before selecting the format with the lowest jitter performance for a given tributary rate.

For each plot, the format exhibiting the lowest predicted worst case jitter performance is indicated by the number (level) of the feasible format. The number of the last and largest feasible format considered is also shown which indicates the total number of feasible formats generated. For example, in Figure 6.5, 62 feasible formats were

generated and the third format exhibited the lowest predicted jitter. The dashed lines between each point on the plots do not convey any specific information but serve to indicate the general trend. Simulation details are the same as those for Figure 6.1b. In particular, recall that the predicted worst case jitter for each feasible format is found within a +/-100 ppm tolerance range of the tributary's rate, in 10 Hz steps starting at the lowest tributary offset frequency. The figures are ordered according to the tributary rate from lowest to highest.

Included in this set of figures are plots showing the intermediate jitter performance of tributary rates used in the experimental hardware for verification and comparison to predicted values. These rates are: 1, 5, 10, 20, and 30 Mb/s, which are all harmonically related to each other and display similar intermediate jitter characteristics; 25.26 Mb/s, which exhibits the highest worst case jitter; and 36.2 Mb/s which is the highest rate tested by the experimental hardware. Other figures show intermediate jitter performance of selected tributary rates whose optimum format sizes or lowest jitter performance vary over a wide range.

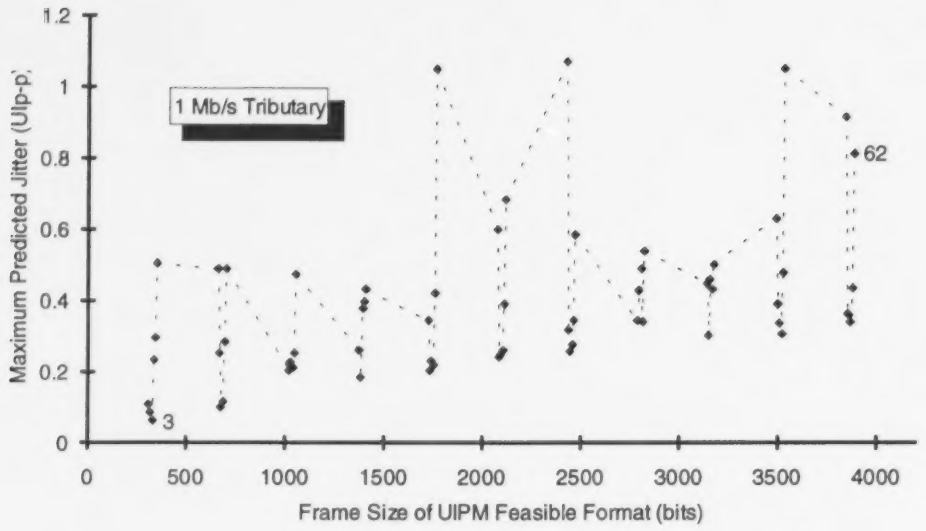


Figure 6.5 Maximum predicted jitter versus format size for 1 Mb/s case

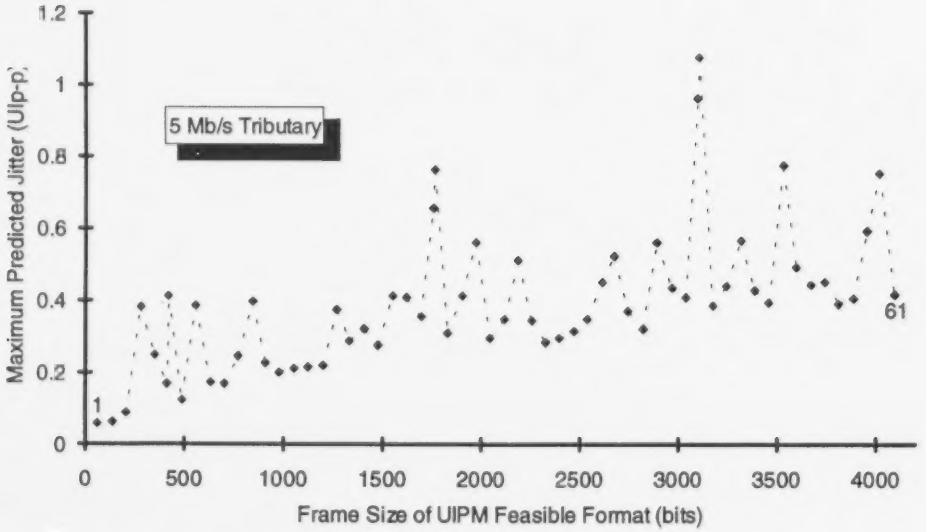


Figure 6.6 Maximum predicted jitter versus format size for 5 Mb/s case

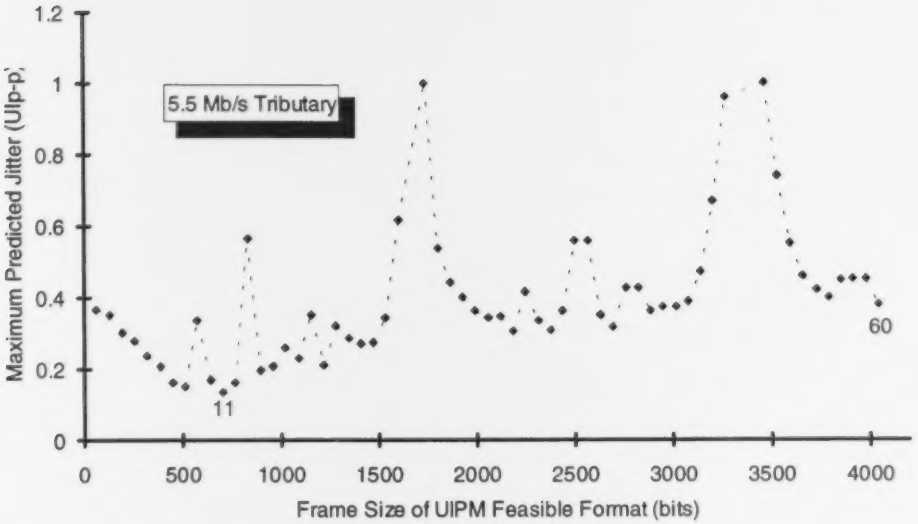


Figure 6.7 Maximum predicted jitter versus format size for 5.5 Mb/s case

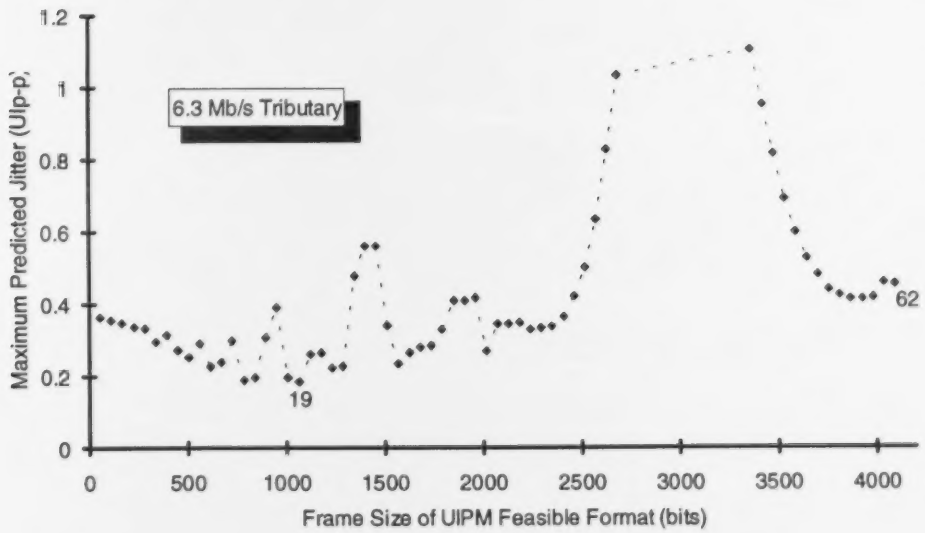


Figure 6.8 Maximum predicted jitter versus format size for 6.3 Mb/s case

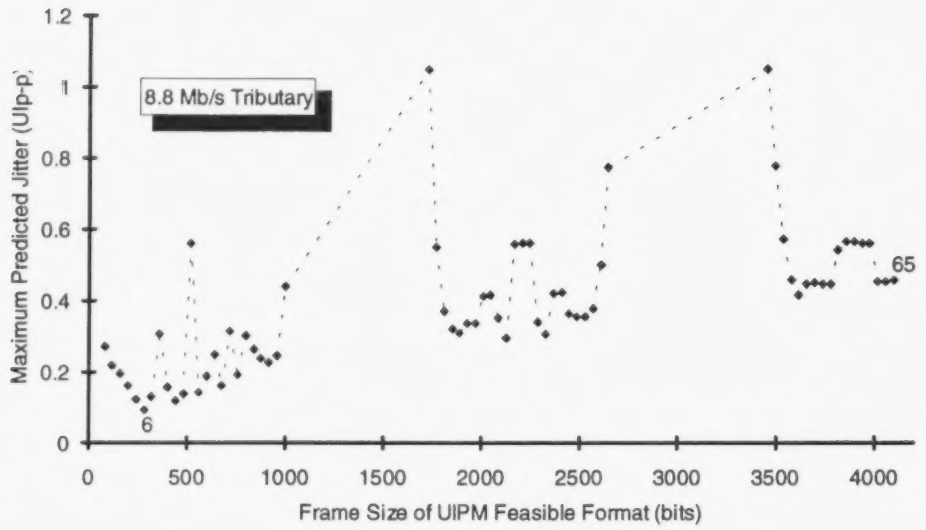


Figure 6.9 Maximum predicted jitter versus format size for 8.8 Mb/s case

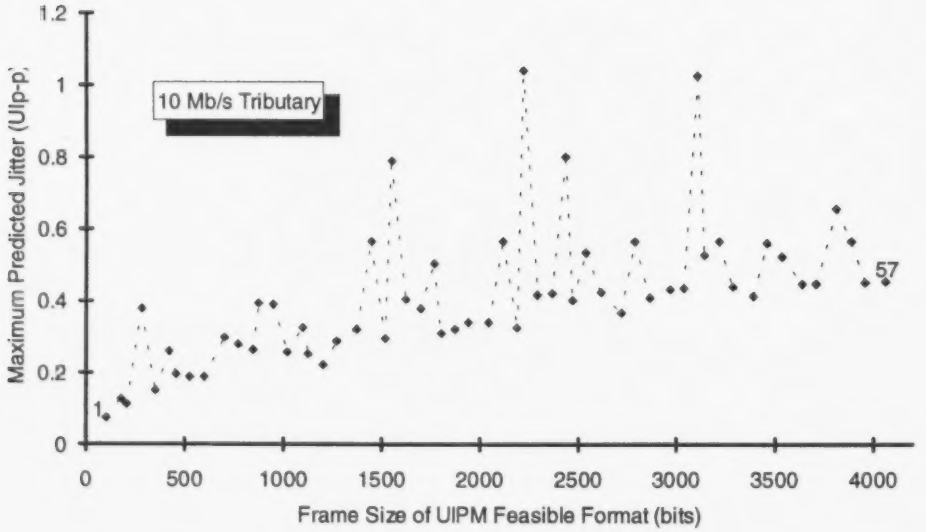


Figure 6.10 Maximum predicted jitter versus format size for 10 Mb/s case

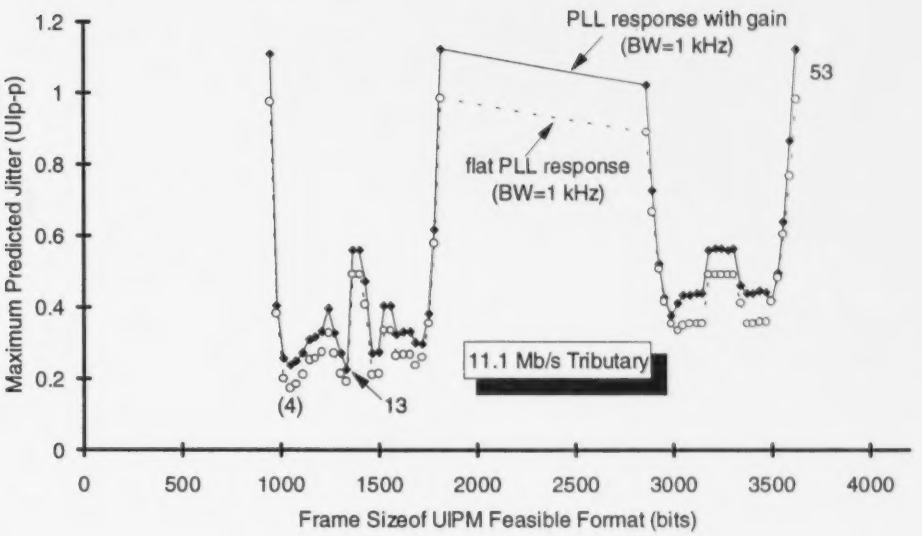


Figure 6.11 Maximum predicted jitter versus format size for 11.1 Mb/s case

In Figure 6.11, two plots are shown. The plot with the solid line joining the points represents jitter simulation using the PLL model described by Equation 4-7 which has some gain in its frequency response (see Figure 4.6). (This is the PLL model used for all the other figures as well.) The plot with the dashed line (and open circles as symbols) represents jitter simulation using the PLL described by Equation 4-6 which has a flat response (see Figure 4.5), except that the bandwidth is retained at 1 kHz. All other simulation details are the same. The purpose for this plot is to explain that the reason some of the above figures show narrowband jitter values that exceed the wideband jitter value of about 1 UI (Figure 6.11 included) is because of the slight resonant gain associated with the frequency response of the PLL model used. The dashed line plot in Figure 6.11 does not contain any NB jitter values that exceed 1 UI, whereas the solid line plot does.

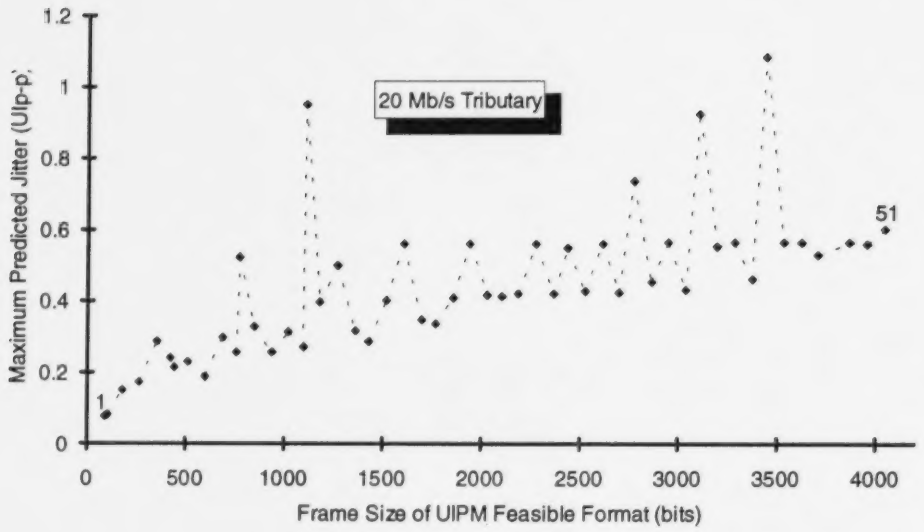


Figure 6.12 Maximum predicted jitter versus format size for 20 Mb/s case

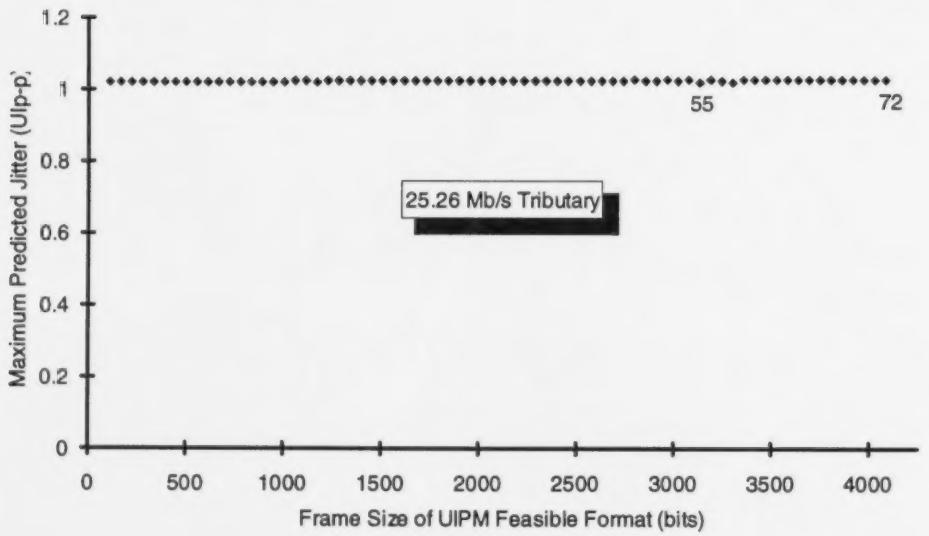


Figure 6.13 Maximum predicted jitter versus format size for 25.26 Mb/s case

Perhaps the most interesting plot is the one shown in Figure 6.13 for the 25.26 Mb/s case. The final UIPM format for this rate had the highest predicted p-p jitter value within the range of tributary rates tested using a resolution of 10 kb/s. It was also the last rate for which the UIPM format design algorithm could generate feasible formats before a no-format region occurred, using a resolution of 1 kb/s (see Figure 6.2). The no-format region contains rates that are approximately 4/7 of the payload rate. A distinguishing feature of Figure 6.13, that is shared to some extent by Figure 6.14 (29.5 Mb/s case), is the flatness the plot - all feasible format sizes in Figure 6.13 exhibit maximum predicted p-p jitter values of about 1 UI.

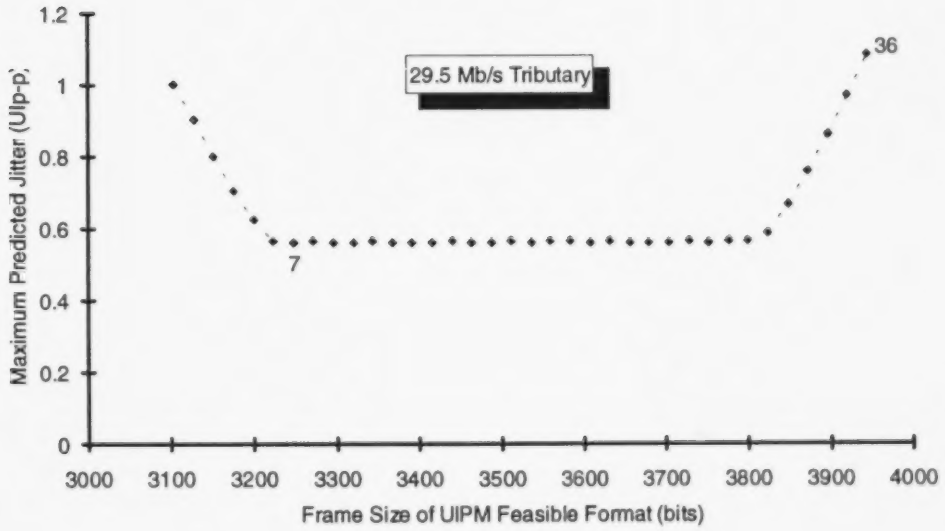


Figure 6.14 Maximum predicted jitter versus format size for 29.5 Mb/s case

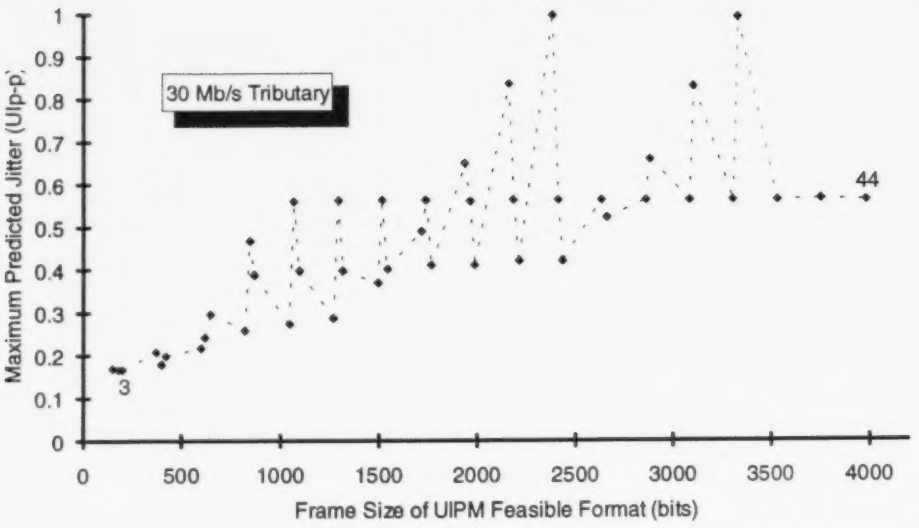


Figure 6.15 Maximum predicted jitter versus format size for 30 Mb/s case

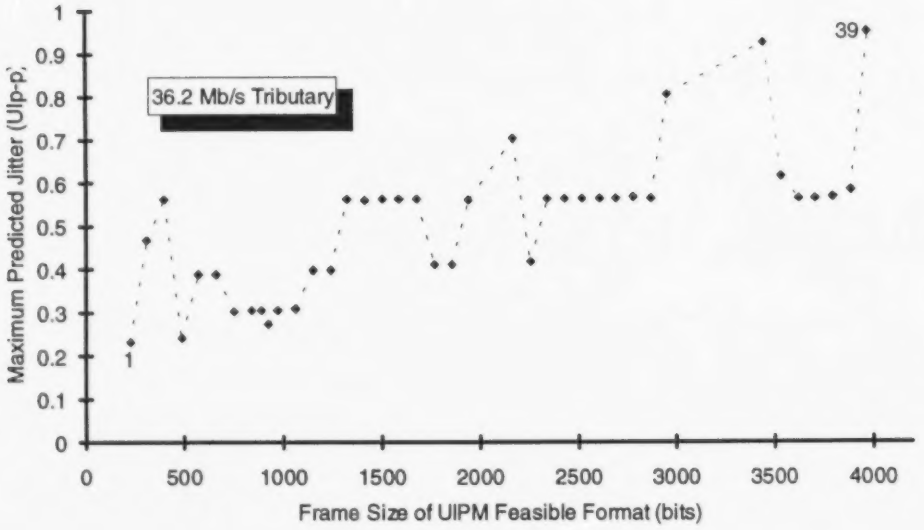


Figure 6.16 Maximum predicted jitter versus format size for 36.2 Mb/s case

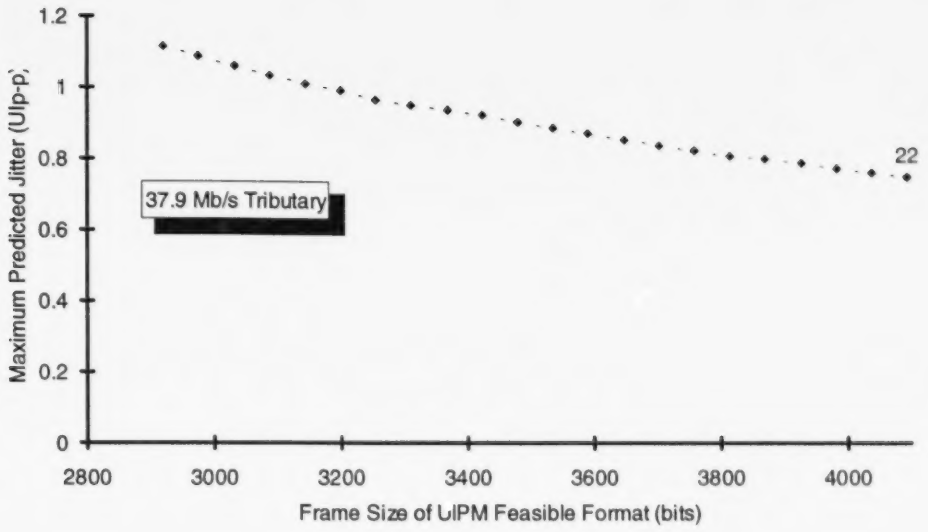


Figure 6.17 Maximum predicted jitter versus format size for 37.9 Mb/s case

Figures 6.13, 6.14, and 6.17 all share the characteristic of fairly regularly spaced points while Figures 6.11, 6.14, and 6.17 share the characteristic of level 1 starting at a relatively large format size. These features are also in contrast to the more "normal" plots (Figures 6.5 to 6.10, 6.12, 6.15, and 6.16) which display characteristics that conform to the general trend shown in Figure 6.1 where larger format sizes result in greater jitter. The explanation for this is that the tributary rates exhibiting these unusual features are all very close to a no-format region where the tributary's entire offset range is very close to one end of the format's offset range (see Figure 6.4). This causes the effective stuff ratio to reach almost 0 or 1.

6.1.3 Selection of Experimental Test Rates

The criteria used to select tributary test rates for comparison between experimental hardware and simulation were:

1. The test rates must correspond to one of the discrete tributary frequencies simulated.
2. The test rates must have UIPM formats small enough to be handled by the experimental hardware.
3. The UIPM formats of the test rates chosen must provide a reasonable range of predicted p-p jitter values.

The tributary test rates selected on this basis were: 1, 5, 10, 20, 25.26, 30, and 36.2 Mb/s. For the 25.26 Mb/s case the smallest format size, rather than the strictly optimum size (see Figure 6.13), was chosen because this allowed experimental testing at this rate and the maximum predicted jitter was virtually identical to that for the optimum size.

Figures 6.18 to 6.24 show the predicted p-p and rms waiting time jitter performance for UIPM formats of the test rates mapped into a DS3 payload. The jitter performance of each format was simulated over a tributary offset range of +/-100 ppm in 10 Hz steps. The UIPM format design algorithm actually performs this offset sweep for each candidate format during its search for the optimum format for any given tributary rate.

The maximum predicted p-p jitter value in the plots of Figures 6.18 to 6.23 correspond to the value determined by the UIPM format design algorithm for selecting an optimum format whose maximum p-p jitter is the lowest among all feasible formats generated for a given tributary rate. An exception is the plot in Figure 6.24 (25.26 Mb/s case) where the actual offset frequencies tested are not identical to the ones determined by the UIPM format design algorithm. This is because the program used to capture the offset sweep adjusts the offset range to ensure a 0 Hz offset is always obtained. The maximum predicted p-p jitter values, along with the corresponding rms jitter values, of the plots in

Figures 6.18 to 6.23 (being optimum formats) appear in the plots of Figure 6.1b, along with jitter values from optimum formats for other tributary rates.

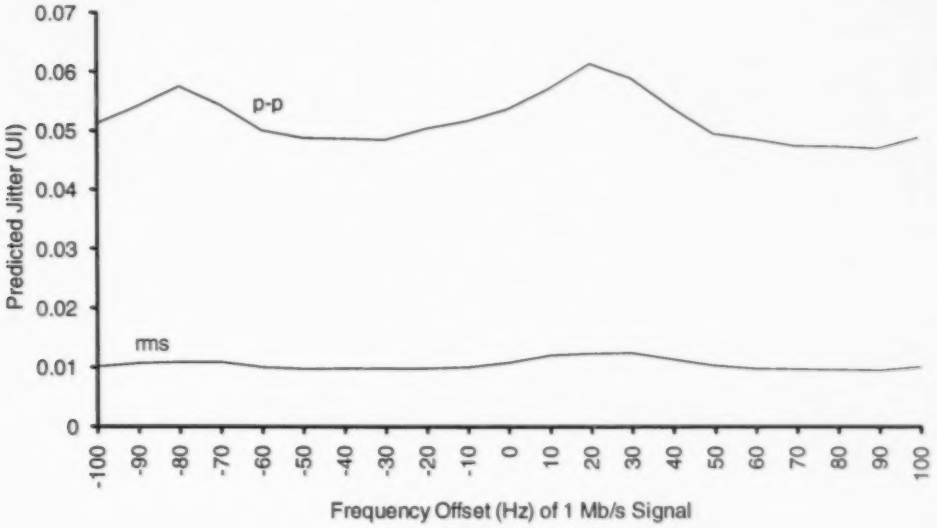


Figure 6.18 Predicted jitter for a 1 Mb/s tributary over a +/-100 ppm offset range (level 3 UIPM format)

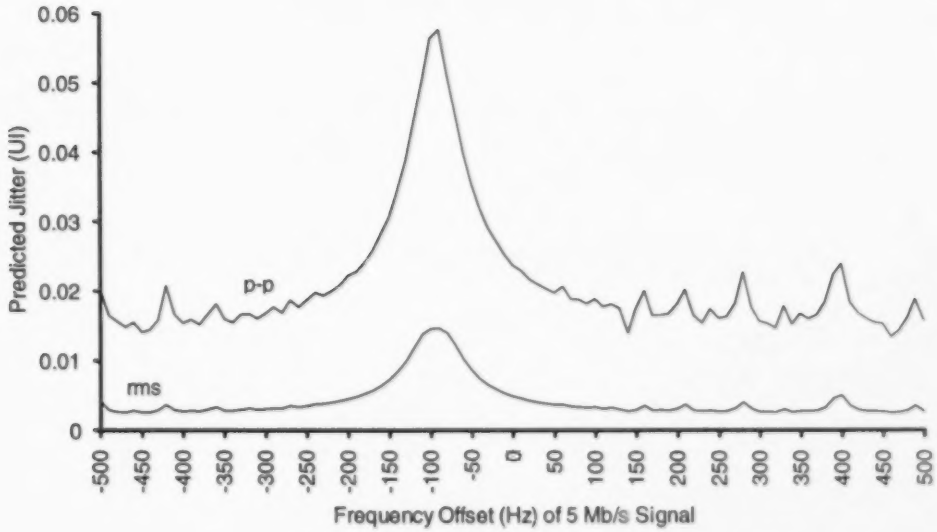


Figure 6.19 Predicted jitter for a 5 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format)

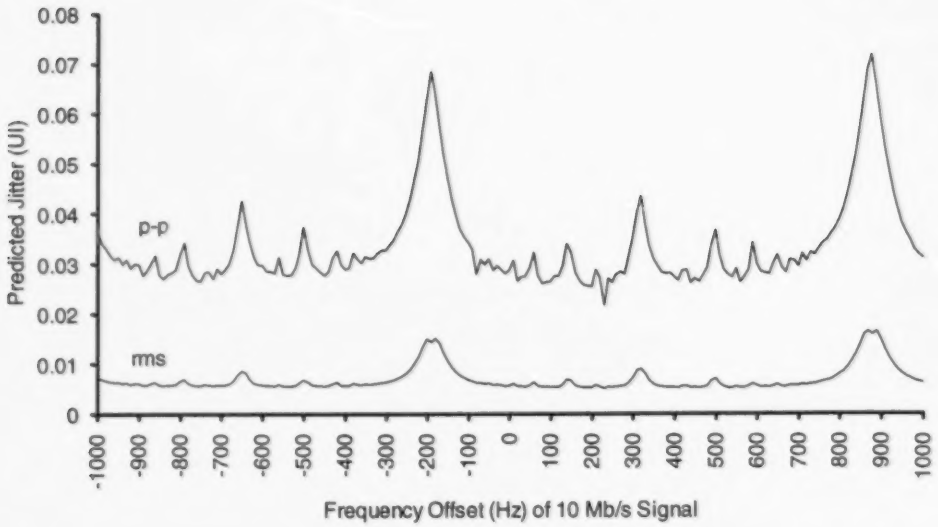


Figure 6.20 Predicted jitter for a 10 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format)

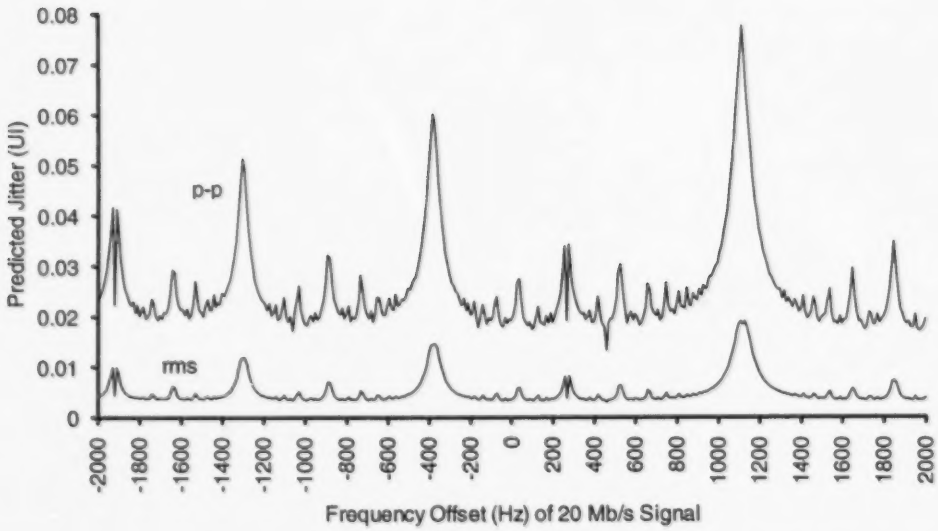


Figure 6.21 Predicted jitter for a 20 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format)

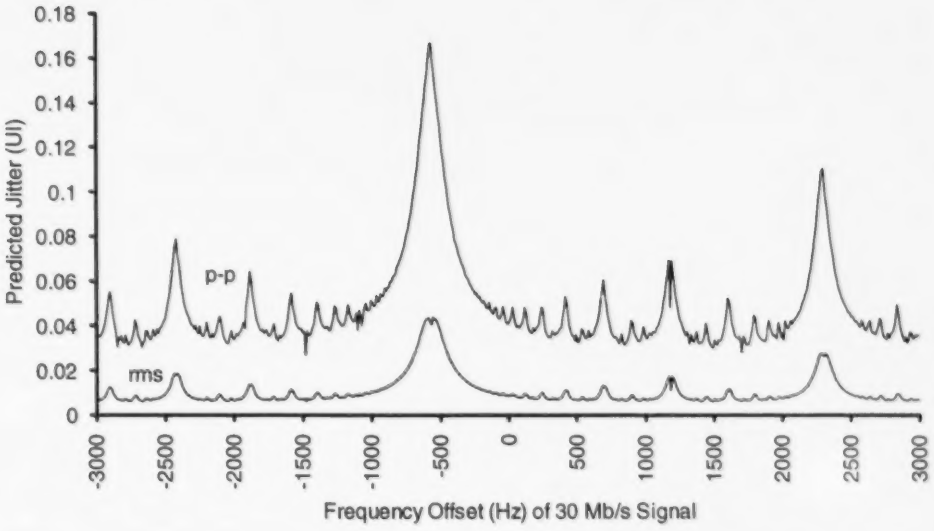


Figure 6.22 Predicted jitter for a 30 Mb/s tributary over a +/-100 ppm offset range (level 3 UIPM format)

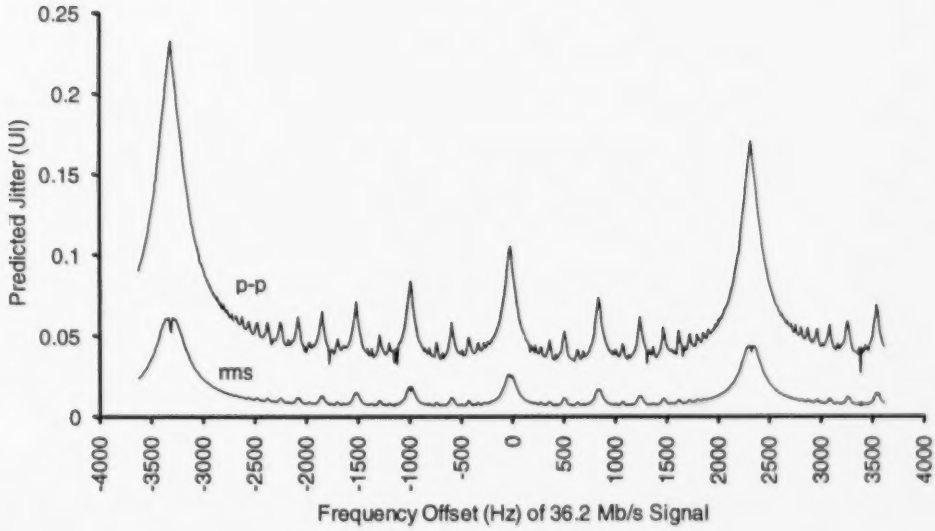


Figure 6.23 Predicted jitter for a 36.2 Mb/s tributary over a +/-100 ppm offset range (level 1 UIPM format)

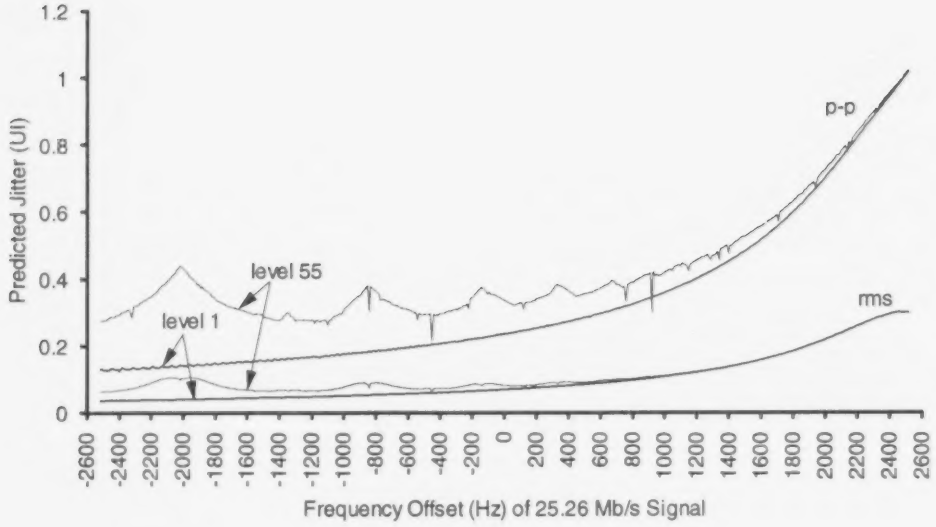


Figure 6.24 Predicted jitter for a 25.26 Mb/s tributary over a +/-100 ppm offset range

The plots in Figures 6.18 to 6.23 show a certain degree of similarity in terms of their general appearance with the resolution of components increasing as the tributary rate increases because a +/-100 ppm tolerance specification was used for all cases. An exception to this similarity is Figure 6.24 where the predicted jitter tends to gradually increase from the negative offset extreme to the positive offset extreme. This is most certainly due to the 25.26 Mb/s tributary rate being positioned at the edge of the x/y=4/7 no-format region.

Two sets of plots are shown in Figure 6.24: one for the selected optimum format size corresponding to level 55, and one for level 1 used for hardware verification. As shown in Figure 6.15, the maximum predicted p-p jitter for these two format sizes are almost the same. In fact, based on Figure 6.24, one could argue that level 1 performance presents a superior format design than level 55 since at 0 Hz offset (for example) the predicted jitter is lower. However, the current version of the UIPM format design

algorithm seeks the lowest worst case predicted jitter anywhere in the combined tributary/carrier offset range, not just near the nominal rate. Note that the worst case jitter occurs at the positive extreme of the +/-100 ppm tolerance range. Actual tributaries operating near this rate may have a smaller frequency offset range in which case the algorithm may select a different level (such as level1) for the optimum format.

Figure 6.25 (same as Figure 4.10) displays the predicted jitter performance of the M12 multiplex format that maps 4 plesiochronous DS1 tributaries into a DS2 synchronous carrier. This figure is included for comparison in jitter performance between a standard telecommunications format and the UIPM formats shown in Figures 6.18 to 6.24.

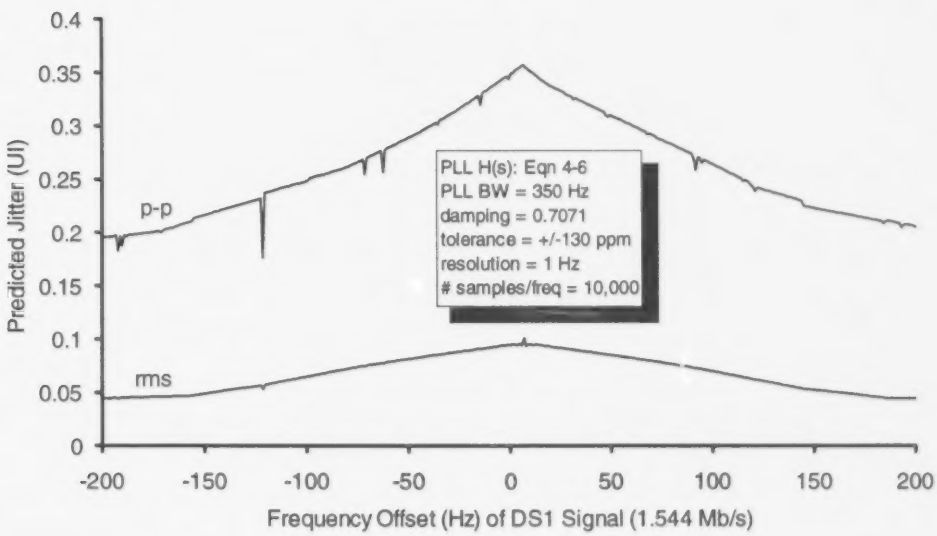


Figure 6.25 Predicted jitter for the M12 case over a +/-130 ppm offset range

6.2 MEASURED PERFORMANCE

The experimental hardware was configured for each of the 7 tributary test rates previously listed and a DS3 payload clock rate of 44.209694 MHz. Each configuration was determined from the UIPM format design parameters for the given test rate and the PLL transfer characteristic matching the experimental hardware.

Table 6.2 details the experimental design parameters for each test rate. This table is divided into 3 sections: UIPM Format Design Parameters, Hardware Configuration Parameters, and PLL Design Parameters. The UIPM format design parameters are outputs of the UIPM format design algorithm. The hardware configuration parameters (discussed in Chapter 5) are manually derived from the format design parameters and are used to set dip switches on the clock gapper board. In the PLL Design Parameters section the value of N is based on the number of data bits per data element (n) for a given UIPM format (usually n+1). The time constants τ_1 and τ_2 (discussed in Chapter 4) are functions of N, the type of VCO circuit (which determines K_v), and the desired jitter attenuation characteristics (i.e. f_c=1 kHz and ζ=1) of the PLL.

Table 6.2 Experimental design parameters

	Test Rate (Mb/s)						
	1	5	10	20	25.26	30	36.2
UIPM Format Design Parameters							
level	3	1	1	1	1	3	1
total bits per frame (T)	328	64	104	88	112	200	224
D bits per subframe (n)	1	1	3	5	8	17	23
R bits per subframe (k)	35	2	5	1	1	3	0
frame rate (f_f, kHz)	134.79	690.78	425.09	502.38	394.73	221.05	197.36
Hardware Configuration Parameters							
# of patterns (P) per data element	1	1	1	1	1	3	1
D bits per pattern (DP)	1	1	3	5	8	5	23
R bits per pattern (RP)	35	2	5	1	1	1	0
extra D bits (DE)	0	0	0	0	0	2	0
extra R bits (RE)	0	0	0	0	0	0	0
PLL Design Parameters							
VCO type	4024	4024	1648	1648	1648	1648	1648
K_v (r/s/v) (K_ϕ=0.5v/r)	$5x10^6$	$2.2x10^7$	$5x10^5$	$1.1x10^7$	$1.1x10^7$	$1.1x10^7$	$1.1x10^7$
N (usually n+1)	8 (≠n+1)	8 (≠n+1)	4	6	9	18	24
τ_1 (ms)	50	140	9.8	140	95	48	36
τ_2 (ms)	0.79	0.79	0.79	0.79	0.79	0.79	0.79

Table 6.3 compares predicted jitter values of the tributary test rates with measurements from the experimental hardware. The predicted results include the nominal stuff ratio (ρ), jitter at the nominal tributary test rate (zero offsets), the minimum and maximum jitter over the tributary's frequency tolerance range of +/-100 ppm, and the respective frequency offsets at which the minimum and maximum jitter values occur. The measured results include jitter at zero offset and jitter at the frequency offsets corresponding to the minimum and maximum predicted jitter.

Table 6.3 Comparison of predicted and measured results

f_t (Mb/s) (+/- 100 ppm)	nominal stuff ratio, ρ	offset (Hz)	Jitter (UI p-p)	
			predicted	measured (+/- 0.007 UI)
1	0.5808	90	min. : 0.047	0.040
		0	nom.: 0.054	0.052
		20	max. : 0.061	0.052
5	0.7618	460	min. : 0.013	0.031
		0	nom.: 0.024	0.045
		-90	max. : 0.058	0.093
10	0.4757	230	min. : 0.022	0.016
		0	nom.: 0.029	0.040
		880	max. : 0.072	0.048
20	0.1897	460	min. : 0.013	0.036
		0	nom.: 0.019	0.032
		1120	max. : 0.078	0.040
25.26 (level 1)	0.0068	-2520	min. : 0.128	0.172
		0	nom.: 0.234	0.308
		2520	max. : 1.019	≥1.0
30	0.2832	1710	min. : 0.028	0.108
		0	nom.: 0.041	0.078
		-560	max. : 0.167	0.150
36.2	0.5832	3390	min. : 0.027	0.076
		0	nom.: 0.092	0.065
		-3310	max. : 0.232	0.156

Measured jitter results were obtained with the desynchronizer PLL output clock displayed in trace-accumulation mode on a digital oscilloscope which was triggered by the original (jitter free) tributary clock. The phase jitter of the recovered clock was allowed to accumulate for a period of 20 seconds before the p-p jitter was measured. As indicated in Table 6.3, the worst case error on the measured values was about estimated to be +/- 0.007 UI. Also note that the measurement for the maximum offset of the 25.26 Mb/s case could not be accurately determined but was ≥1 UI p-p.

Figure 6.26 presents a graphical comparison between the predicted and measured results recorded in Table 6.3. Although the jitter values measured at the frequency offsets corresponding to the minimum and maximum predicted jitter are not always the smallest and largest respectively, there is a good agreement between predicted and measured results. Certainly the measured results are in the same order of magnitude as the predicted results. This indicates that the UIPM format design algorithm is a reliable basis for format design and that implementations can in practice achieve the jitter levels predicted for UIPM designs.

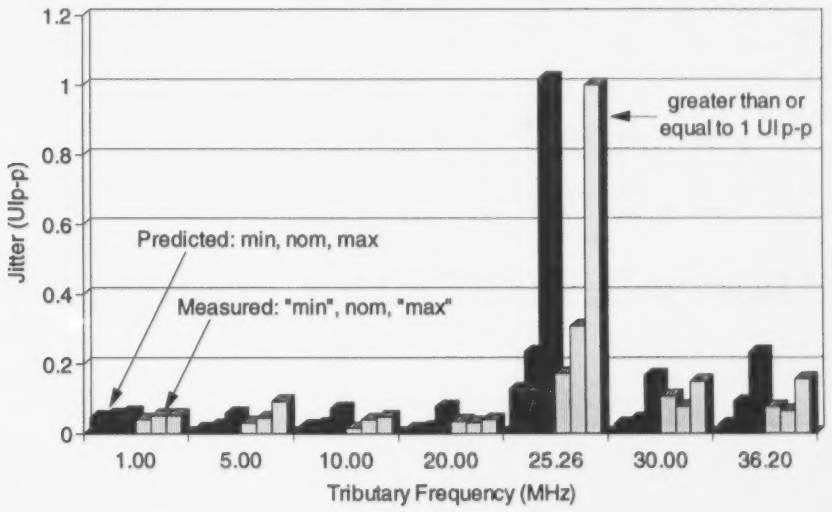


Figure 6.26 Graphical comparison of predicted and measured results

In addition to the tributary test rates listed in Table 6.2, measurements were performed on the experimental hardware over the entire +/-100 ppm operating range of the 10 Mb/s and 25.26 Mb/s cases to validate predicted jitter levels of simulated offset sweeps. Figures 6.27 and 6.28 show the predicted and measured jitter for these cases. Figures 6.27 and 6.28 are the same as Figures 6.20 and 6.24 respectively except that measured results are overlaid on top of the predicted results.

The +/-100 ppm tolerance range corresponds to a frequency offset of +/-1000 Hz for the 10 Mb/s signal and +/-2520 Hz (rounded to the nearest 10 Hz increment) for the 25.26 Mb/s signal. Predicted jitter values were obtained for offset increments of 10 Hz, while measured jitter values were taken at increments of 50 Hz for the 10 Mb/s tributary and 100 Hz for the 25.26 Mb/s tributary. The raw measured results were subjected to a three point moving average to clarify their trend.

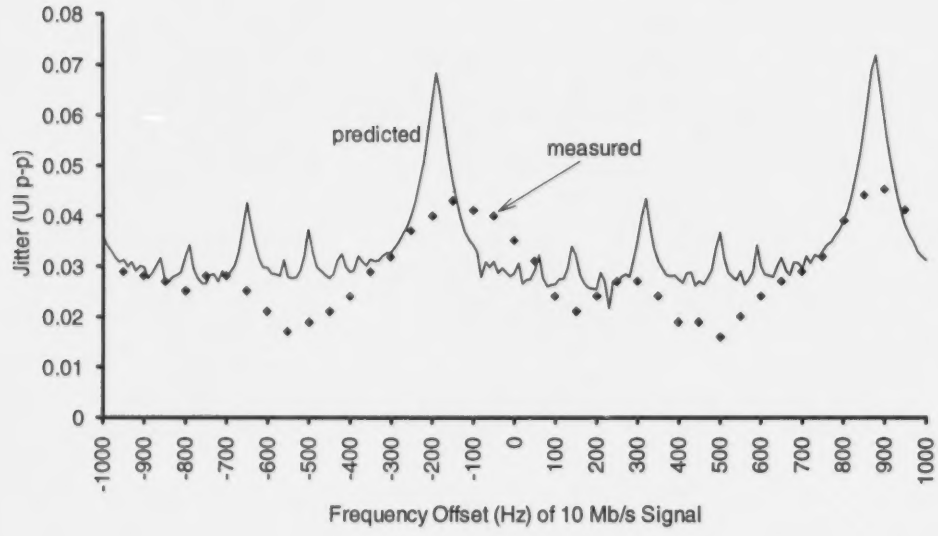


Figure 6.27 Comparison of predicted and measured jitter for a 10 Mb/s case

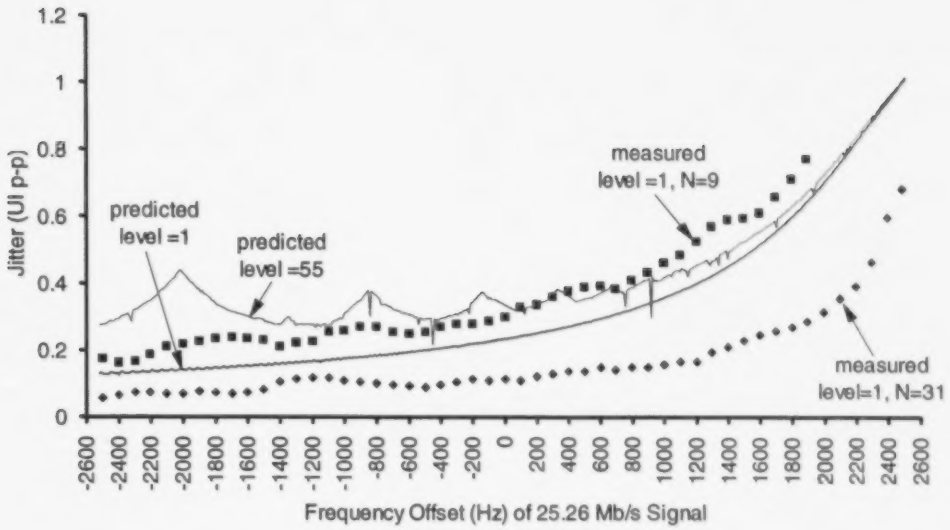


Figure 6.28 Comparison of predicted and measured jitter for a 25.26 Mb/s case

For the 25.26 Mb/s offset sweep shown in Figure 6.28, measured results were obtained for divide-by-N values of 9 and 31. The N=9 value was based on n+1 while N=31 was found to result in optimum hardware performance for this case. (Values of N greater than 31 resulted in degraded performance because control over VCO stability began to suffer.) It can be seen in Figure 6.28 that measured results with N=9 are closer to predicted results than measured results with N=31. It was also observed that even values of N resulted in noticeably worse jitter performance (even N=30 caused increased jitter) for the 25.26 Mb/s case.

The results in Figures 6.27 and 6.28 show relatively good agreement between predicted and measured jitter values. Differences between the predicted and measured results are attributed to inaccuracies in computer modeling of the exact transfer function of the experimental desynchronizer PLL and intrinsic phase noise of the PLL circuit.

One aspect of the difference between predicted and measured results shown in Figure 6.27 is similar to the smearing effect that input jitter has on waiting time jitter

generation described in [BLab82], in that the peaks and valleys of the output jitter occurring over an offset range become eroded and smoothed out. The combined effect of slight modeling inaccuracies and prototype phase noise may thus be responsible for this aspect of the measurements in that they have an effect equivalent to jitter on the tributary input.

Figures 6.29 to 6.35 show actual waveforms from the experimental system corresponding to each tributary test rate and a DS3 payload carrier. In each figure, the top trace is the original tributary clock while the bottom trace represents the recovered tributary clock after the PLL. The indicated p-p jitter on the recovered clock consists of waiting time jitter (WTJ) due to the synchronization/desynchronization process plus a measured amount of intrinsic jitter from the PLL system itself. For example, in Figure 6.31a the 6.0 ns p-p jitter of the 10 Mb/s tributary consists of 2.0 ns p-p intrinsic jitter plus 4.0 ns p-p or 0.04 UI p-p of jitter due to the multiplexing process alone. Figure 6.31b shows the 2.0 ns p-p intrinsic jitter generated by the PLL operating with the original tributary clock as its input rather than the gapped DS3 payload clock.

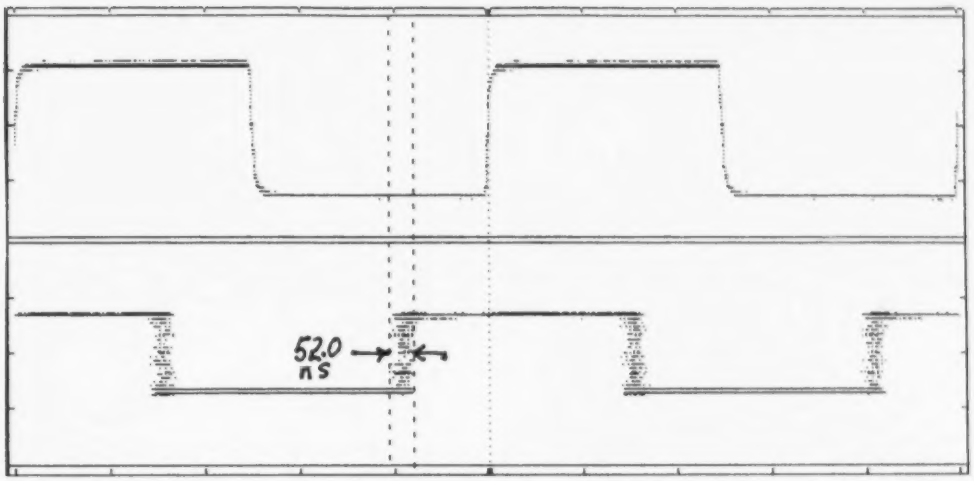


Figure 6.29 Original (top) versus recovered (bottom) tributary clock for 1 MHz case

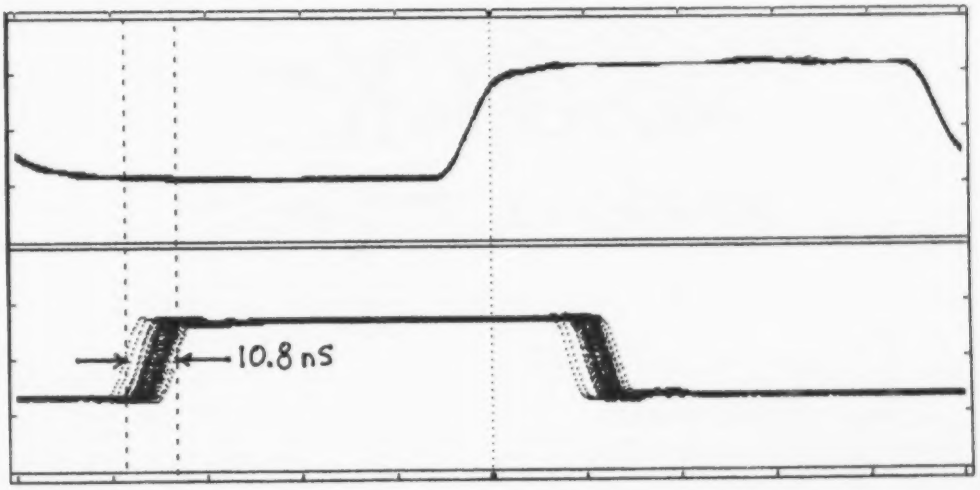
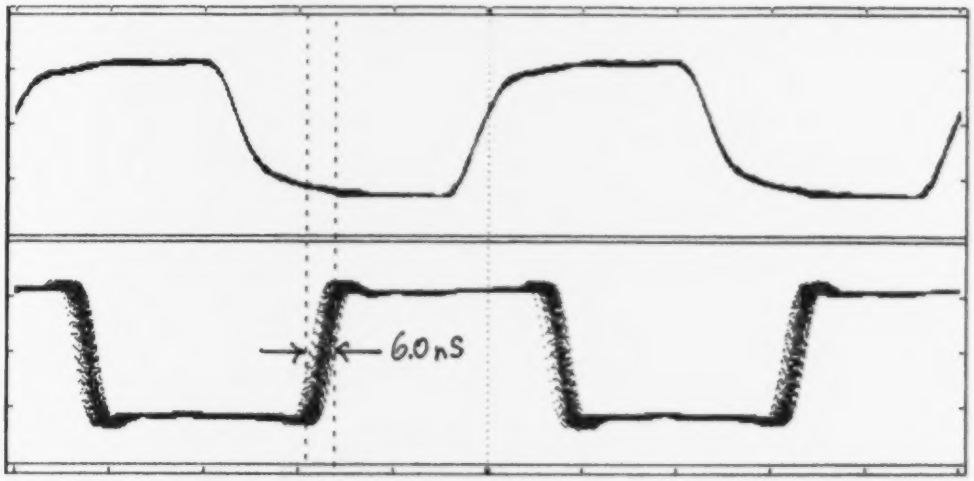
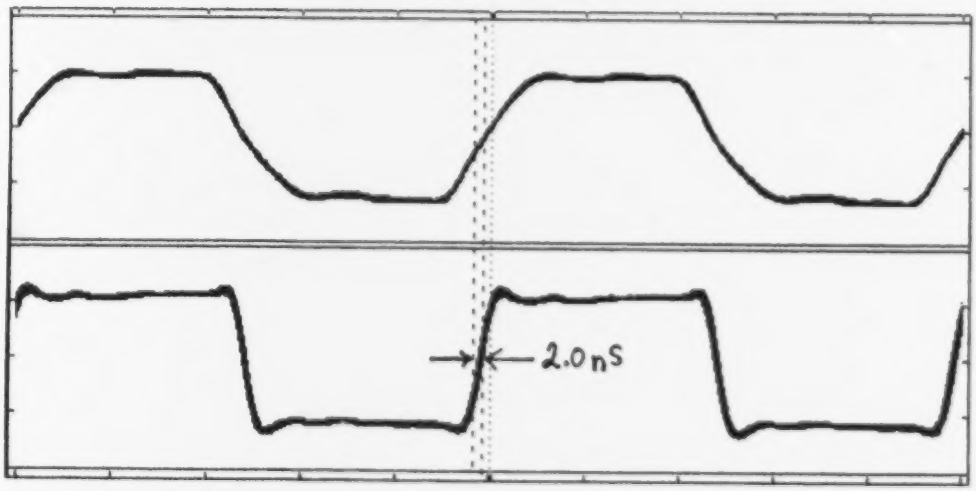


Figure 6.30 Original (top) versus recovered (bottom) tributary clock for 5 MHz case



a) Original (top) versus recovered (bottom) tributary clock



b) Top: original tributary clock. Bottom: output of PLL with original tributary clock as input, showing intrinsic jitter

Figure 6.31 Waveforms for 10 MHz case

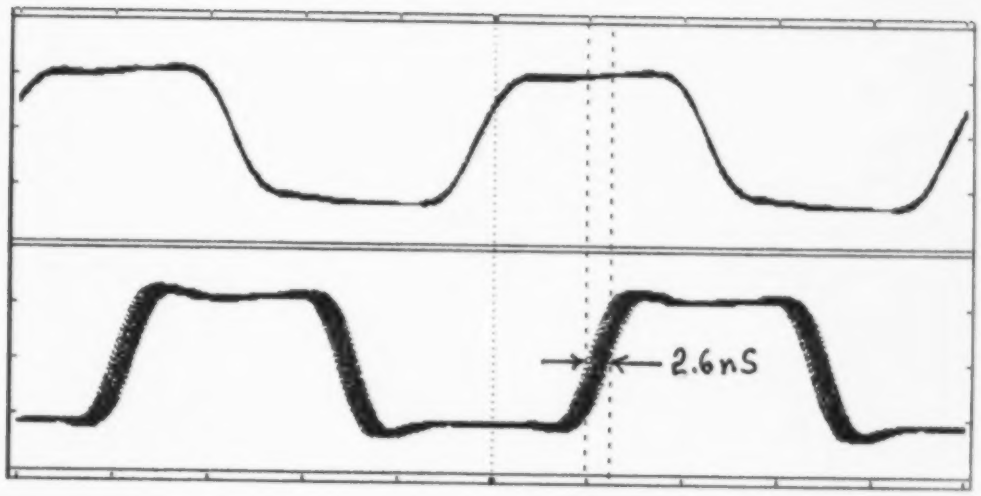
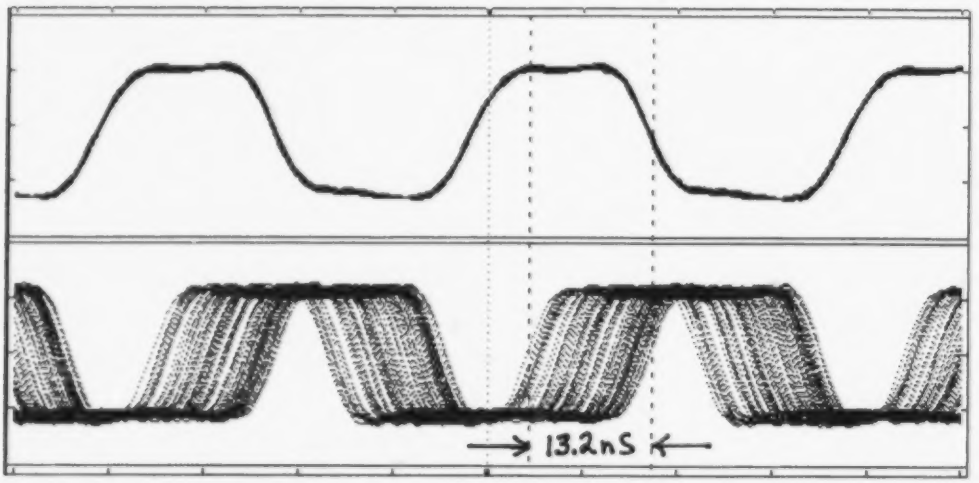
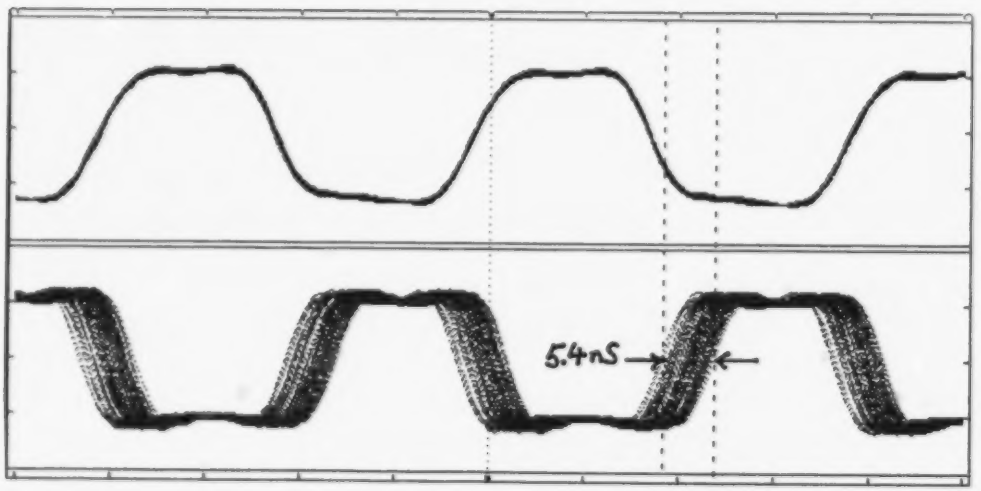


Figure 6.32 Original (top) versus recovered (bot) tributary clock for 20 MHz case



a) Original (top) versus recovered (bottom) tributary clock, with N=9



b) Original (top) versus recovered (bottom) tributary clock, with N=31

Figure 6.33 Waveforms for 25.26 MHz case

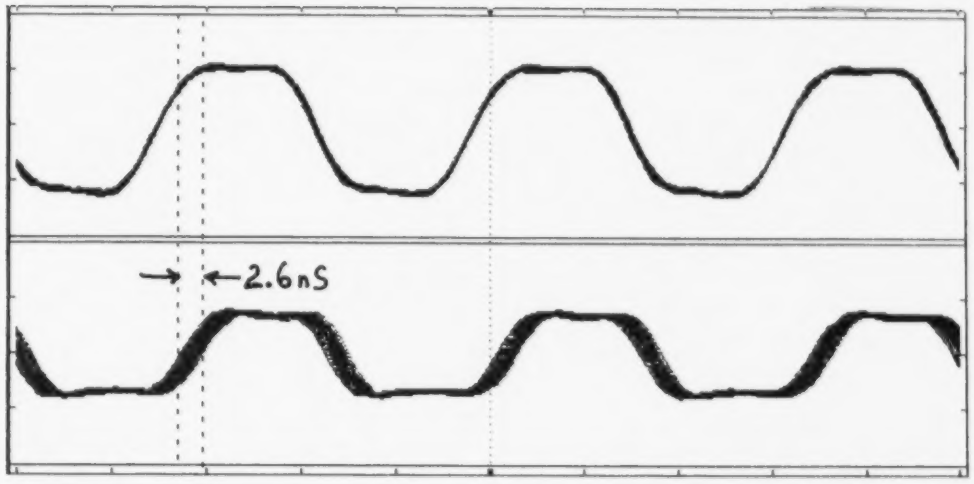


Figure 6.34 Original (top) versus recovered (bot) tributary clock for 30 MHz case

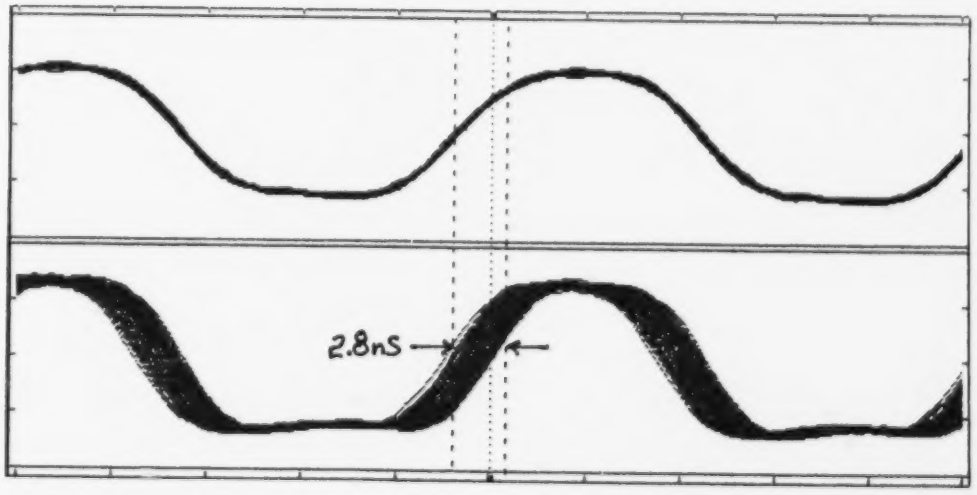


Figure 6.35 Original (top) versus recovered (bot) tributary clock for 36.2 MHz case

6.3 SUMMARY

The worst case waiting time jitter simulated as part of the UIPM format design algorithm was plotted for a range of tributary rates adapted into a DS3 payload rate of 44.209694 Mb/s. The range extended from 1Mb/s, increasing in increments of 100 kb/s, to 44 Mb/s. All tributary rates had an offset range of +/-100 ppm. Results shown in Figure 6.1b show the majority of rates caused worst case predicted jitter of well under 0.5 UI p-p. This result is significant because it suggests that transport of arbitrary-rate digital signals over the telecommunications network via UIPM technology is possible.

Predicted results with finer resolutions of increments over limited ranges indicated regions of no-format rates where the UIPM format design algorithm was unable to generate any feasible formats. Further investigation revealed that the regions of no-format rates occur when $f_t \approx f_p(x/y)$ where $x \in \{1,2,3,...,7\}$ and $y \in \{x+1,...,8\}$. For these no-format rates the frequency tolerance criteria could not be met. Also, the fact that the denominator of the fraction x/y never exceeds 8 is directly related to there being 8 subframes per UIPM frame. Despite the existence of no-format rates, more than 99% of all tributary rates from the minimum possible rate of 75.414 kb/s to the maximum possible rate of 43.77422 Mb/s (in 1 b/s increments) could be accommodated by the UIPM format design algorithm.

A general analysis of the predicted results shows that the jitter tends to increase as the tributary rate approaches the payload rate. This is because the frame size must increase to maintain adequate data capacity, causing f_{so} to decrease and jitter to subsequently increase. However, when one extreme of the tributary tolerance range is near the format tolerance boundary (e.g. tributary rates near no-format regions), a larger format may result in better jitter performance by moving the effective stuff ratio away from a 0 or 1 condition, even though f_{so} is lower.

Figures 6.5 to 6.17 show plots of worst case predicted jitter versus format size for various tributary rates. These plots offer insight to the decision process used by the UIPM format design algorithm for determining an optimum format for each case. The most interesting plots involve tributary rates that are close to no-format regions for which there is little flexibility remaining in the format design.

A number of tributary test rates were chosen to compare predicted and experimental results. The test rates chosen represented a sample of the entire possible range, had format sizes small enough to be handled by the experimental hardware, and provided a reasonable range of predicted p-p jitter values. These rates were: 1, 5, 10, 20, 25.26, 30, and 36.2 Mb/s. The jitter performance of each rate was simulated over an offset range of +/-100 ppm in 10 Hz steps. These offset sweeps were plotted (Figures 6.18 to 6.24) to provide a prediction of how each UIPM format would perform over the operating range of each tributary test rate.

The experimental hardware was used to obtain p-p jitter measurements of each test rate at 0 offset and offsets corresponding to the minimum and maximum predicted p-p jitter. Relatively good agreement was shown to exist between predicted and measured results (Figure 6.26). In addition to the measurements discussed above, measurements were performed on the experimental hardware over the entire +/-100 ppm operating range of the 10 Mb/s and 25.26 Mb/s cases in offset increments of 50 Hz and 100 Hz respectively, and plotted on top of the corresponding predicted offset sweeps (Figures 6.27 and 6.28). Again, relatively good agreement existed between the predicted and measured set of results.

The predicted jitter performance of UIPM format designs and results from experimental hardware presented in this chapter serve to verify the UIPM concept and demonstrate the feasibility of its implementation. Chapter 7 presents recommendations and suggestions for further study and future implementations of UIPM. This includes the

development of a complete 1/1 UIPM system and issues regarding n/1 and n/m UIPM systems.

7 FUTURE WORK

The UIPM system discussed throughout this thesis is a Phase 1 concept dealing only with one tributary mapped into one standard carrier. The experimental hardware itself, used for proof-of-concept and predicted performance verification, encompasses only part of a complete Phase 1 UIPM system. This chapter presents recommendations and suggestions for further study and future implementations of UIPM based on analyses of predicted and measured results of the UIPM experimental hardware.

Future work involving the UIPM system naturally falls into three related areas: software, hardware and generic format structure. The area of software concerns itself with further development and refinement of the UIPM format design algorithm for Phase 1 and subsequent phases of the UIPM concept, ultimately involving multiple mixed-rate tributaries and multiple carriers. The area of hardware concerns itself with development of existing and additional hardware to complete a Phase 1 UIPM prototype system and eventually a UIPM system capable of accommodating a multiple-tributary/multiple-carrier scenario. Finally, modifications to the UIPM generic format structure may result in enhanced performance of a Phase 1 UIPM system and subsequent UIPM systems. All three areas of the UIPM system are inter-related to the extent that development in one area will necessitate further development in the other two areas.

7.1 UIPM GENERIC FORMAT STRUCTURE

Issues involving further development of the UIPM generic format structure include:

- final definition of the O_m, O_d, O_r, and E bits
- modifications resulting in improved Phase 1 jitter reduction and increased flexibility enabling carriage of all possible tributary rates
- necessary modifications to address requirements of subsequent UIPM phases

Section 7.1.1 deals with the first point while Section 7.1.2 deals with the remaining two points.

7.1.1 Definition of the Subframe Overhead Bits

The O_m, O_d, O_r, and E bits in the generic format structure are defined in Chapter 3 in terms of their purpose but not in terms of their implementation. In other words, it has not yet been determined how these bits will convey their required information. The O_m bit is reserved for future use and could be used to indicate the number of frames within the master frame for carriage of multiple tributaries. The coding of the M bits in a multiple tributary format structure may differ from one situation to another depending on the number of frames per master frame. This information could also be conveyed by the O_m bits.

The purpose of the O_d and O_r bits is defined for a single frame format structure. In a multi-frame format structure each frame could have an independent configuration of D and R bit distribution given by the O_d and O_r bits. In the UIPM experimental hardware the information conveyed by these bits are manually entered into the P, DP, RP, DE, and RE divider blocks.

The E bits are defined for error detection and correction. Implementation of these bits has not been determined, although some type of forward error correcting code may be feasible with the E bits acting as distributed checklist. There are 8 E bits per frame regardless of the format size. Therefore there may be constraints on the correcting capability as format sizes increase. For multi-frame format structures, the E bits could be used on a per frame basis or for the entire master frame.

7.1.2 Other Generic Format Modifications

Any modifications to the existing UIPM generic format structure must be done with more flexible n/m UIPM systems in mind so that compatibility exists between the

various possible phases of UIPM system development since this translates into various grades of service. Expansion of the UIPM generic format structure to include multiple tributaries and/or multiple carriers has already been built in. Therefore, additional modifications need only focus on issues such as enhanced jitter performance and flexibility.

Chapter 6 discusses the existence of relatively narrow no-format regions that center around tributary rates which are specific fractions of the carrier payload rate. These fractions are related to the number of subframes in one frame of a UIPM format. Although the total fraction of no-format rates comprise less than 1% of all possible tributary rates (in 1 b/s increments) that can be carried by the UIPM generic format, using a generous tributary frequency tolerance of +/-100 ppm, it may be possible to reduce or eliminate this amount by modification to the generic format structure itself. This has already been proven by manipulation of the carrier's stuff bit to adjust f_p, although this does not affect the structure of the UIPM generic format.

Tributary rates near no-format regions exhibit relatively high predicted worst case jitter values as discussed in Chapter 6. Therefore, reduction or elimination of no-format regions would result in fewer tributary rates with comparatively poor jitter performance. One method of accomplishing this would be to reduce the number of subframes per frame of the UIPM generic format since this would reduce the set of fractions that describe the relationship between the carrier payload rate and no-format tributary rates. This also increases the flexibility of the generic format structure because the minimum format size increment in bits is equal to the number of subframes.

Any reduction in the number of subframes in the generic format structure must, however, ensure that certain format characteristics are retained such as equal spacing between the F bits. In the current generic format structure three C bits are used to indicate the status of the S bit since this allows effective one bit error correction among the C bits. With the presence of one M bit, four F bits are then required for an equi-distance framing pattern. Since the UIPM generic format already contains error control

bits, one possible modifications is to reduce the number of subframes to six by having only three F bits and two C bits. The disadvantage of this modification is that the framing pattern is less robust because there is an increased chance of UIPM payload data mimicking the framing pattern, increasing the probability of misframes. Reduction of the generic format structure to four subframes containing only two F bits and one C bit is also possible, but framing is even less robust and an additional method of determining start of frame is necessary.

One final modification suggestion is to use two (or more) S bits instead of just one within each frame. To do this, one of the data bits is replaced with an S bit. Thus the total number of bits does not change but the number of dedicated data bits has decreased. The effect is that the format can now accommodate a lower tributary offset than before. This would reduce the number of no-format rates that fail to meet the frequency tolerance criteria, thus increasing the flexibility of the UIPM generic format structure and reducing the number of high jitter formats. The addition of another S bit would require three additional C bits if C-bit triplication is desired, thus increasing the number of subframes. However, because the UIPM generic format already has error control bits, only two C bits in total may be necessary to indicate the status of the two S bits.

7.2 UIPM FORMAT DESIGN ALGORITHM

Issues involving further development of the UIPM format design algorithm include:

- modifications to the format selection process to include efficiency considerations in multiple tributary designs
- incorporation of a method to determine appropriate trade-off between format size and jitter performance

- exploitation of variable numbers of the S bits in a given standard carrier format to resolve no-format conditions
- possible use of stuff threshold modulation [GrMM87] to resolve high jitter cases
- modifications to accommodate subsequent UIPM phases.

The first two points are discussed in Section 7.2.1, the next two in Section 7.2.2, and the last one in Section 7.2.3.

7.2.1 Efficiency, Format Size, and Jitter Performance

Efficiency, format size, and jitter performance are related issues when considering enhancement modifications to the UIPM format design algorithm. The issue of efficiency was discussed in chapters 3 and 4. For a 1/1 UIPM system, use of the carrier payload could end up being quite inefficient. Small format sizes are generally less efficient than large format sizes because of the ratio of data bits to total bits in one UIPM frame. However, smaller format sizes tend to exhibit better jitter performance (i.e. less waiting time jitter) than larger format sizes. Smaller format sizes are also easier to implement in the current experimental hardware prototype, although this limitation is technology dependent and not algorithmically dependent.

As pointed out in Chapter 4, the issue of efficiency may be a moot point since a realistic UIPM system would utilize as much carrier payload bandwidth as possible by multiplexing a number of related or non-related tributaries into the payload of one or more carriers. For a 1/1 UIPM system, efficiency could be increased by pre-multiplexing harmonically related tributaries into a higher rate tributary, but this may result in a large format that exhibits relatively high jitter.

If the issue of efficiency is indeed irrelevant, then possible modifications to the UIPM format design algorithm need only address format size and jitter performance. There are cases (such as the 25.26 Mb/s case) where the optimum format exhibited worst case predicted jitter that was only marginally less than other suitable format sizes. If the

hardware is sensitive to format size, then a format size smaller than optimum, but with only slightly high jitter, may be a better choice. Also, some non-optimal formats suitable for a particular mapping scenario may have their worst case predicted jitter near the extremes of the tributary offsets and exhibit lower jitter at nominal rates than the format size selected as optimum.

Although no specific modification suggestion have been given in this section, the above discussion serves to highlight areas of improvement for future consideration.

7.2.2 Other General Modifications to the Algorithm

Other modifications to the UIPM format design algorithm appear as additions rather than changes. One addition already investigated in Chapter 6 involves manipulation of the S bits that exist as part of the carrier's format structure. Aside from the existing plesiochronous digital hierarchy formats (DS1, DS3, etc.), the only other synchronous transport method in the telecommunications industry is SONET which also incorporates variable stuff bits. The S bits in a carrier's format are manipulated by considering one or more S bit as part of the carriers payload. This has the effect of changing the payload rate slightly. The number of available carrier S bits could simply be an additional input parameter for the UIPM design algorithm and a set of suitable formats would be created for each carrier payload rate. Manipulation of carrier S bits would likely have a greater effect on hardware modifications than on software modifications.

Incorporation of stuff threshold modulation [GrMM87] is another addition to the UIPM format design algorithm that could be employed to reduce jitter of high jitter formats. This technique varies the phase difference threshold in the synchronizer of a multiplex system that determines if a stuff should occur during the next available stuff opportunity. It has been shown in [GrMM87] that a significant reduction in jitter occurs as a result of stuff threshold modulation. This technique could easily be implemented in

the UIPM format design algorithm and would also require appropriate modifications to the UIPM hardware.

One additional modification to the format design algorithm that could be considered concerns the relationship between the PLL cutoff frequency (f_c) and the stuff opportunity rate (f_{so}). It is desirable to ensure that f_c is much lower than f_{so} so that the frequency component at f_{so} is greatly attenuated within the passband of the modeled PLL and does not distort the simulated waiting time jitter values. This feature could be incorporated into the algorithm by stating that feasible format sizes must have an f_{so} (equivalent to frame rate for current UIPM generic format structure) that is greater than f_c by some specified amount. Although this criteria was not actually implemented in the current version, a study of Table 6.2 shows that the frame rate of the UIPM formats selected for the test rates are at least 100 times the f_c of 1 kHz.

7.2.3 Algorithm Modifications for n/m UIPM Systems

Remaining modifications to the UIPM format design algorithm relate to future n/m UIPM systems. In general, the algorithm must be modified and expanded to select among suitable formats that combine a number of mixed-rate tributaries multiplexed together into a single carrier payload or multiple carrier payloads. Accomplishing this would probably require the most work of all modifications to the UIPM format design algorithm suggested thus far.

The most practical way to implement an n/m UIPM system is to have m n/1 UIPM sub-systems with pre-selection hardware that determines which inputs will be associated with a particular carrier. For each sub-system, f_p is divided among the tributaries in fractions determined by the algorithm. Hence a faster tributary may get a larger percentage of f_p. This method is developed further in Section 7.3.4.

Figure 7.1 shows an example of a multi-frame UIPM generic format adapting three tributaries into a single carrier. Subscripts following M and F bits indicate possible

framing codes while those following D and S bits correspond to a particular tributary. The double subscripts following the C bits indicate tributary and control bit number, in that order. Each subframe contains overhead bits followed by bit (as in Figure 7.1) or byte interleaved tributaries multiplexed with R bits. The actual number of data bits from each tributary, the number of R bits, and the distribution of D and R bits must be determined by the format design algorithm. Figure 7.1 does not depict an actual design case but serves to illustrate the multi-frame concept.

M_1 [OH {D_1 D_2 D_3 R...}] F_1 [ ] C_{11} [ ] F_0 [ ] C_{12} [ ] F_0 [ ] C_{13} [ ] F_1 [S_1]
M_0 [OH {D_1 D_2 D_3 R...}] F_1 [ ] C_{21} [ ] F_0 [ ] C_{22} [ ] F_0 [ ] C_{23} [ ] F_1 [S_2]
M_1 [OH {D_1 D_2 D_3 R...}] F_1 [ ] C_{31} [ ] F_0 [ ] C_{32} [ ] F_0 [ ] C_{33} [ ] F_1 [S_3]

Figure 7.1 An example of a multi-frame UIPM generic format
(OH = O_m, O_d, O_r, E)

There are additional constraints the UIPM format design algorithm must deal with at the master frame level. For example, as shown in Figure 7.1, it is impractical to have each frame carry single tributary data because very large buffers would be necessary for each frame, adding extra delay and complexity. Also, although each frame of the master frame can have a unique distribution of D and R bits, all subframes in the master frame must have the same number of bits so that the framing bits are equi-distant.

7.3 COMPLETE UIPM SYSTEM HARDWARE

Much of the existing UIPM experimental hardware can be modified or reused for development of a complete UIPM system. Therefore, Section 7.3.1 discusses problems and recommendations associated with the existing UIPM experimental hardware. Subsequent sections discuss details of the components necessary for completion of a Phase 1 UIPM prototype system, while Section 7.3.4 addresses issues regarding n/1 and n/m UIPM systems.

7.3.1 Existing UIPM Experimental Hardware

Problems associated with the experimental hardware can be categorized into those related to the clock gapper and those related to the PLL. The only real problem encountered with the clock gapper circuitry was clock skew due to varying propagation delay of the various subcircuits. The solution was to use a number of delay elements in the clock distribution path, although this did not solve all clock skew problems. For the implementation of the tributary test cases, however, the problem of clock skew was overcome. For use in a complete UIPM system, a more practical solution would be to implement the clock gapping and other UIPM multiplexing/demultiplexing logic circuitry in VLSI where propagation delays are much less.

Problems in the experimental hardware associated with the PLL center around the high intrinsic jitter associated with the large-ratio VCO design. This problem was addressed by using two small-ratio VCO circuits exhibiting low intrinsic jitter, at the expense of manually changing circuit components within these VCOs and within the analog circuitry (filter time constants) for each tributary rate. If intrinsic jitter were not a problem, use of the large-ratio VCO would allow clock recovery of any tributary rate within a range of about 1-50 Mb/s without any component switching in the VCO module. In a production UIPM system this would be desirable.

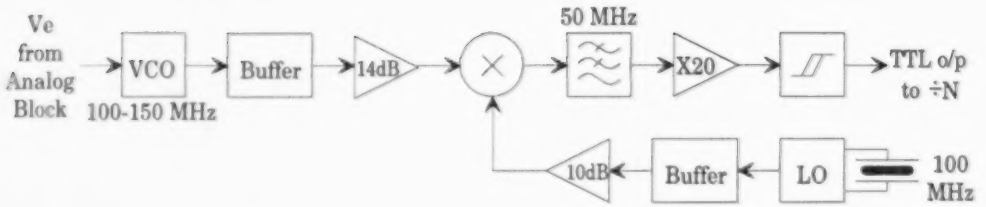


Figure 7.2 Large-ratio VCO block diagram (1-50 MHz range)

Figure 7.2 shows the block diagram of the large ratio VCO module while complete schematics are given in Appendix E. In the large-ratio VCO circuit a small ratio 100-150 MHz VCO (tuning ratio of 1.5 to 1) is mixed with a 100 MHz crystal controlled local oscillator (LO). The output of the mixer is filtered to attenuate the sum frequency and harmonics and only pass the difference frequency. This results in a frequency range of 0 to 50 MHz over the input voltage range of the small ratio VCO. The output of the filter is then amplified and squared up to provide a TTL output. Figure 7.3 shows the transfer function of the wide ratio VCO. The conversion gain (K_v) of this VCO is approximately $4x10^7$ rads/s/volt.

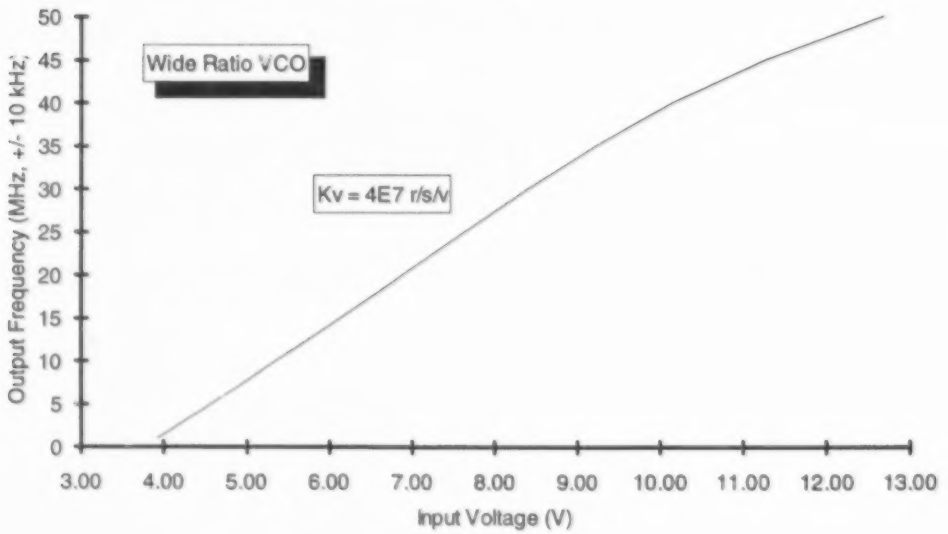


Figure 7.3 Transfer function of the wide ratio VCO

The main reason for the excessive phase noise in the large-ratio VCO is residual harmonics after the LPF. This appears as a small amount of amplitude and phase noise superimposed on the difference frequency signal and results in phase noise after the amplifier and squaring circuit. To reduce the intrinsic jitter of the large-ratio VCO a number of suggestions are presented. One suggestion is to replace the passive mixer in the VCO module with an active one such as the MC1596 or with a non-switching type mixer. This may create less harmonic distortion resulting in less phase noise. The LPF following the mixer could also be changed to provide a steeper cutoff with greater attenuation of the harmonics (the main culprit). For example, use of a high order elliptical filter may offer considerable improvement if zeros of the filter are placed at the frequencies of the offending harmonics. Although the phase characteristic of the elliptical filter is very nonlinear, this would not affect its operation as a loop filter because the filter output is essentially a DC voltage. Alternatively, a comb filter following the LPF would also serve to notch out the harmonic frequencies. The circuitry following the LPF could also be simplified by using a line receiver IC instead of an amplifier and several stages of signal squaring, thus reducing extra phase noise.

An additional suggestion for improving the PLL system concerns enhancements to the analog block. For obtaining experimental results, the analog block consisted of a PD/LPF combination. This required calculating the LPF time constant τ_1, and physically changing the associated filter components, for each experimental case because of the dependency of τ_1 on N (Equation 5-6). This was considered a minor disadvantage for obtaining conceptual verification but would not be desirable for a production UIPM system. A method to eliminate the dependency of τ_1 on N is shown in Figure 7.4. Here an *N-compensator* is added to the PD/LPF circuit of the analog block to normalize the effect of the changing ÷N in the PLL transfer function.

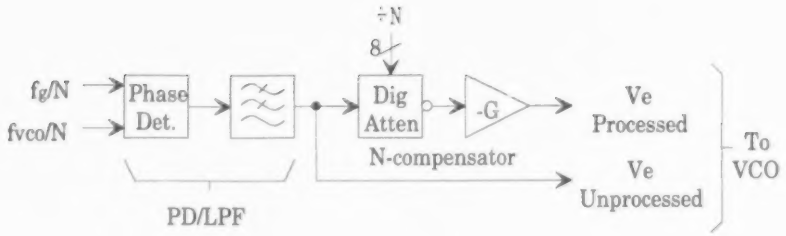


Figure 7.4 Analog block with N-compensator

The N-compensator consists of a digital to analog converter (DAC) connected as a digital attenuator and a variable gain amp (-G). The digital attenuator combined with the variable gain amp maintains a constant ratio of A/N where A is an additional gain variable in the expression for K resulting in $K=AK_\phi K_v$, rather than $K=K_\phi K_v$ as before. This allows the value of τ_1 in Equation 5-6 to remain constant when N changes. Without this measure, the PLL transfer function would change each time N is changed (a disadvantage that was ultimately accepted during the experimental phase). The digital attenuator uses a DAC whose digital input comes from the divide-by-N block. Thus as N increases the attenuation decreases, or conversely, the overall gain increases.

The variable gain amp following the digital attenuator partially compensates for the loss of the digital attenuator. The absolute gain of the digital attenuator, using an 8 bit DAC, is N/256 resulting in a maximum gain of essentially unity with N=255 (all bits high). Including the gain of the amp (G) the overall gain is A=G*N/256 resulting in A/N=G/256 (eliminating the effect of N). It is not necessary to have A/N=1 (i.e. G=256) as long as the loop gain ($AK_\phi K_v/N$) is much greater than the natural frequency ω_n [Best84]. Therefore G is set to the largest value possible that does not add excessive noise to the system. Since the output of the digital attenuator is actually inverted, the amp is connected in an inverting mode (hence negative G) to restore the polarity.

The N-compensator was actually included as part of the original experimental hardware for use with the large-ratio VCO, which has a constant K_v. Since two small-

ratio VCOs were used, however, K_v did not remain constant, but τ_1 is also dependent on K_v and the whole point is to maintain a constant τ_1. Hence the N-conpensator circuitry was not used. Appendix E shows the schematic of the analog block including the N-compensator.

Another consideration is to change the overall design of the PLL. One possible solution would be to design a wide range (if possible) all-digital PLL eliminating the analog and RF circuitry susceptible to noise and temperature variations. Another possibility shown in Figure 7.5 retains the general analog/digital hybrid design of the classical PLL but addresses the conflicting design requirements of maximum jitter attenuation (narrowband operation) and control of the VCO's stability (wideband operation). This conflict is most prevalent when dealing with large-ratio VCOs.

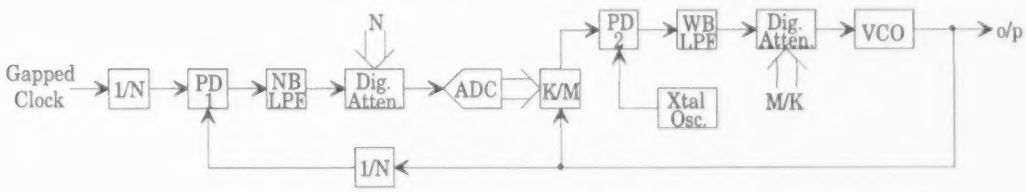


Figure 7.5 Alternate PLL design

In Figure 7.5, the VCO module consists of another PLL configured as a frequency synthesizer using a highly accurate and stable crystal oscillator as its reference. The frequency synthesizer PLL uses a relatively wideband (WB) LPF so that the stability (intrinsic phase noise) of the VCO is controlled and is locked to the crystal oscillator. The frequency is controlled by the K/M block whose digital input is derived from the error voltage of the main PLL through an analog-to-digital converter (ADC). The K/M block is a fractional divider that allows the VCO's frequency to be changed in finer increments than possible with just an ordinary divider. Of course, the VCO in this design changes in discrete frequency steps thus adding some small amount of intrinsic jitter. The main PLL

uses a narrowband (NB) LPF for jitter attenuation which does not affect the performance or stability of the VCO.

As shown in Figure 7.5 both PLL circuits use a digital attenuator to maintain constant system characteristics over the range of the divider inputs. This design also requires a large-ratio VCO of some sort, but the advantage is control over the VCO stability while allowing considerable jitter attenuation. Details of this design have not been developed and actual performance is unknown. However, it is one idea that may ultimately result in a wide range self adjusting PLL design for a real UIPM system.

7.3.2 Phase 1 UIPM Multiplexer

Figure 2.17 shows the block diagram of a Phase 1 UIPM system consisting of a multiplexer and demultiplexer. The experimental hardware for this thesis consisted only of clock path components including the stuffing mechanism, clock gapping logic, and desynchronizer PLL. For a complete Phase 1 UIPM system data path components are also required.

Figure 7.6 shows a more detailed block diagram of a Phase 1 UIPM multiplexer. The function of the frequency analysis block shown in Figure 2.17 is more conceptual than actual, and is to determine the frequency and tolerance of the tributary clock. This block does not appear in Figure 7.6 because its function is replaced by manually entering this information into the UIPM format design algorithm.

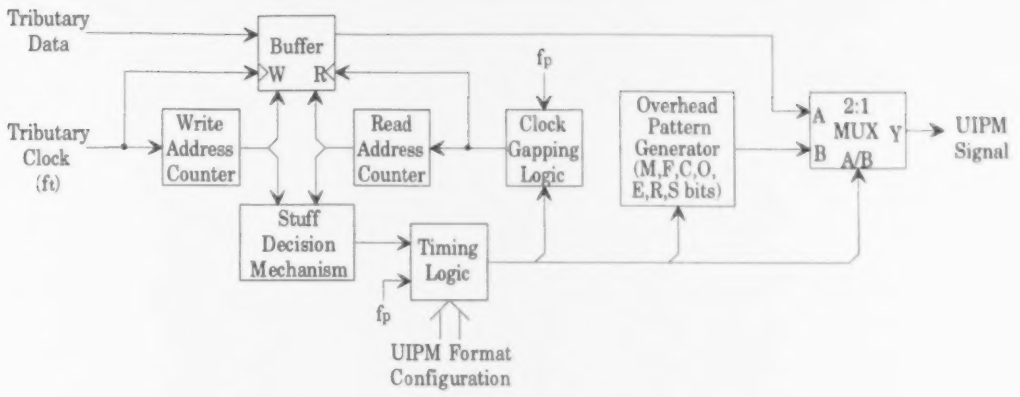
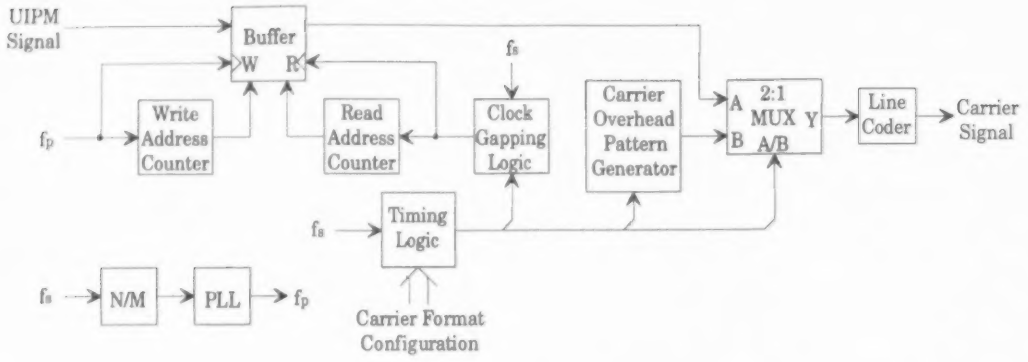


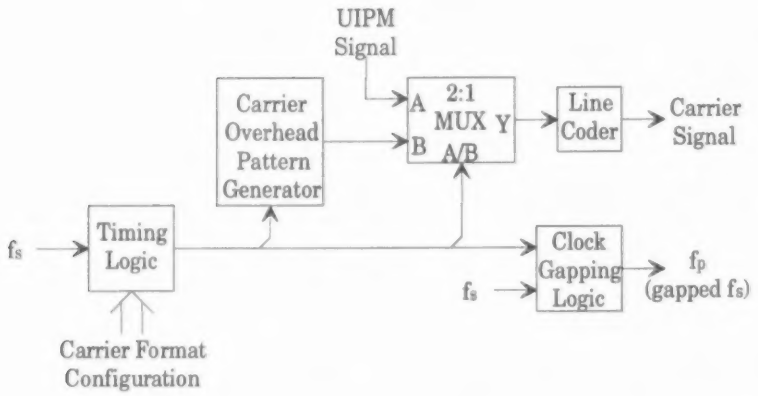
Figure 7.6 Phase 1 prototype of UIPM multiplexer

The tributary data is written into the addresses of the buffer memory at the speed of the tributary clock (f_t). This data is read out of the buffer at the speed of the gapped payload clock. The payload clock (f_p) is gapped whenever a UIPM format overhead bit needs to be inserted and whenever the S bit is stuffed with a dummy bit. The stuff decision mechanism monitors the pointers of the read and write address counters and determines if the next available S bit should carry a dummy instead of a data bit in order to prevent the read address counter from passing the write address counter. The stuff decision mechanism used in the experimental hardware can be reused as is for the Phase 1 UIPM prototype multiplexer and already contains the read and write counters.

The timing logic block produces all the necessary timing and clock signals for the multiplexer. This block contains programmable counters and dividers that are configured by manually entered inputs determined by the UIPM format design algorithm. The overhead pattern generator produces the necessary UIPM overhead bits, which include the M, F, C, $O_{m,d,r}$, E, R, and S bits, at the appropriate times in the bit stream. The overhead bits are multiplexed with the data bits to form the UIPM signal.



a) method 1



b) method 2

Figure 7.7 Adding carrier overhead to a UIPM signal

Figure 7.7 shows two methods of multiplexing the UIPM signal with carrier overhead to create a standard carrier signal. Method 1 in Figure 7.7a is the most straight forward method conceptually but the most complex practically. In this method the carrier payload clock (f_p) is assumed to be a true periodic clock signal at the carrier payload rate. To derive f_p from the carrier ciock (f_s), f_s must first be divided by N/M which is the ratio of carrier payload bits to total bits in the carrier format (e.g. N/M=84/85 for DS3). The output of the N/M block is a gapped clock whose average rate is the carrier payload rate.

If this gapped clock is passed through an appropriate PLL the output is a periodic clock signal whose rate is equal to the carrier payload rate.

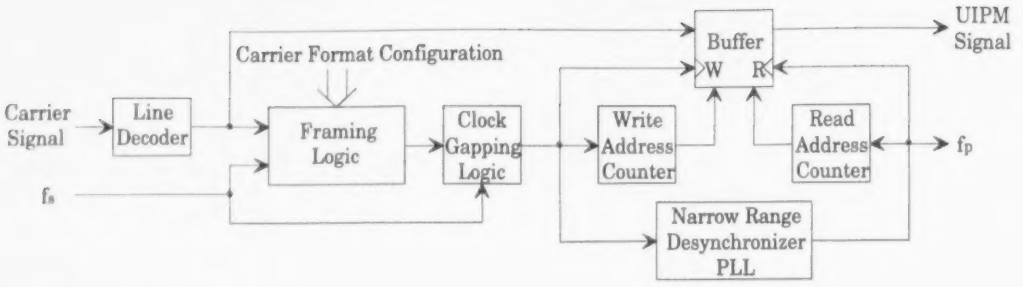
The process of multiplexing the UIPM signal with carrier overhead is virtually identical to the method of multiplexing tributary data with UIPM overhead as shown in Figure 7.6, except for a few differences. A stuffing mechanism is not necessary because the UIPM signal fits exactly into the payload portion of the carrier signal. Also, a line coder is added to the end of the process to convert the carrier bit stream (e.g. NRZ) into the proper physical signal (e.g. B3ZS for DS3) for transmission on a given media.

A simpler but less intuitive method conceptually for achieving the addition of carrier overhead to the UIPM signal to create a standard carrier signal is shown in Figure 7.7b (method 2). Here the payload clock consists of the gapped carrier clock and the gaps correspond to the time slots in the carrier signal where carrier overhead bits will exist. Therefore the tributary data will be read out of the buffer in the UIPM multiplexer at the carrier clock rate most of the time, but not read when gaps are present. During these gaps a carrier overhead bit will be inserted into the UIPM bit stream.

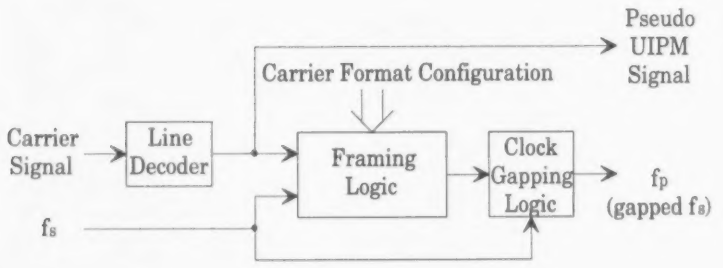
Since the average frequency of the gapped carrier clock is equal to the carrier payload clock - used as an input parameter to the UIPM format design algorithm - the stuffing mechanism and timing logic should function properly and the tributary data should be correctly multiplexed according to the UIPM format design.

7.3.3 Phase 1 UIPM Demultiplexer

The first stage in the UIPM demultiplexer is recovery of the UIPM signal from the carrier signal. Again, two methods are presented for achieving this and are shown in Figure 7.8. The method shown in Figure 7.8a is the more conceptually straight forward solution but is also more complex than the method shown in Figure 7.8b. Both methods implement a line decoder to convert the physical carrier signal into a normal binary bit stream.



a) method 1



b) method 2

Figure 7.8 Demultiplexing a UIPM signal from a standard carrier

In method 1, the framing and clock gapping logic cause only carrier payload bits (UIPM bits) to be read into a buffer and effectively removes the carrier overhead bits. The clock gapper gaps the carrier clock edges corresponding to carrier overhead bits. The average frequency of the gapped carrier clock is derived by the narrow range PLL and the resulting output is the carrier payload clock which is phase synchronous to the original carrier payload clock in the UIPM multiplexer. The carrier payload clock is used to read the bits out of the buffer and this bit stream is the recovered UIPM signal. This process is essentially the reverse of the process depicted in Figure 7.7a.

Method 2 shown in Figure 7.8b simply provides a gapped carrier clock as the payload clock and the decoded carrier signal as the UIPM signal. In this method, the significant edges of the carrier payload clock (gapped f_s) correspond only to UIPM bits in

the carrier signal (pseudo UIPM signal). In the main UIPM demultiplexer circuitry the gapped carrier clock will ensure that no carrier overhead bits are read into the buffer memory.

Figure 7.9 shows the main UIPM Phase 1 demultiplexer system for recovery of tributary clock and data. The operation of the UIPM demultiplexer is virtually identical to the method shown in Figure 7.8a for recovery of the UIPM signal. The only difference is that a line decoder is not required and that the desynchronizer PLL performs over a wide frequency range. Additional detail of the framing and stuff control logic block and the clock gapping logic block appear in Figure 7.9.

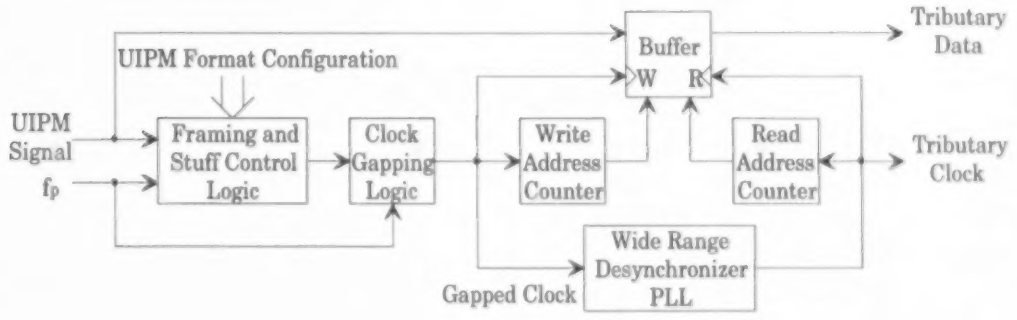


Figure 7.9 Phase 1 prototype of UIPM demultiplexer

The purpose of the framing and stuff control logic block is to provide timing signals that indicate the location of the various UIPM overhead bits in the UIPM bit stream and to indicate the status of the S bit. This information is passed to the clock gapping logic block which gaps the carrier payload clock during time slots associated with UIPM overhead bits. To provide this information framing must be established and continually monitored to ensure that only tributary data bits are read into the buffer.

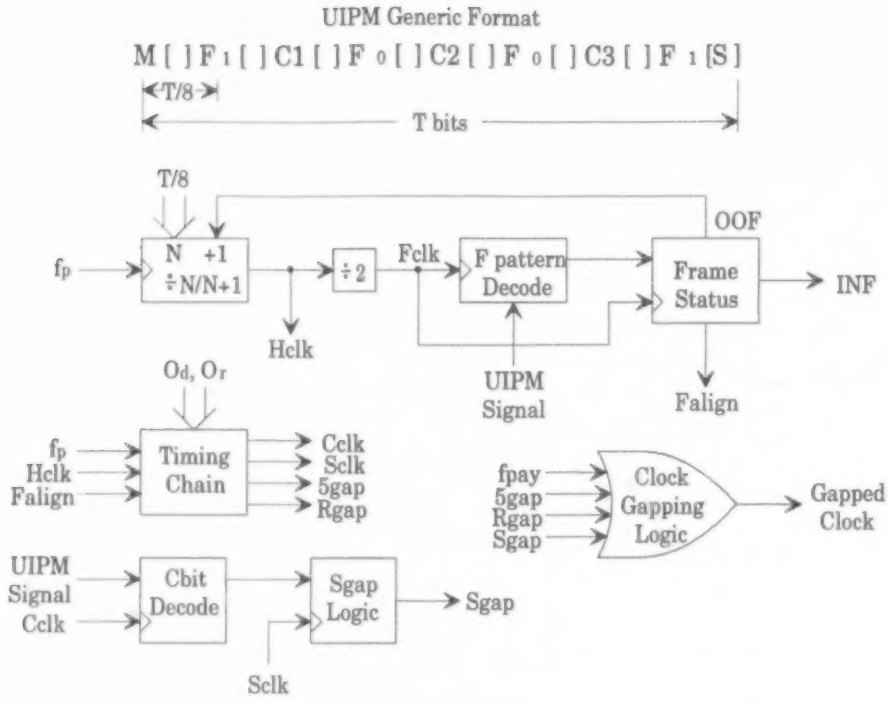


Figure 7.10 Framing and clock gapping logic in a UIPM demultiplexer

Figure 7.10 shows the UIPM generic format with a framing pattern of 1001 (which is the same as the DS3 framing pattern). With this type of framing pattern, framing can be established without any additional information. For multi-frame formats additional information such as M bit coding would be required to indicate start of master frame. The number of bits from one major UIPM overhead bit to another (outside a subframe: [ ]) is T/8 where T is the total number of bits in the frame. The carrier payload clock is divided by T/8 in the N/N+1 divider block to produce the housekeeping clock (Hclk) whose significant edges occur at the same rate as the major overhead bits.

Alignment of Hclk with the major overhead bits is accomplished by searching for the F bit framing sequence of 1001. The Hclk is divided by 2 to produce the Fclk which is used to decode assumed F bits in the bit stream. The frame status block determines if the decoded F bit sequence is valid. If the decoded sequence is not 1001 then the value of N

in the divider block is incremented by 1 for one period of f_p which is equivalent to halting f_p for one period. This shifts the alignment between Hclk and the bit stream by one bit and the framing process is repeated until a valid framing sequence is detected.

A confirmation process in the frame status block is applied which declares in frame (INF) status only after a consecutive number of valid framing words have been detected and declares out of frame (OOF) status only after a consecutive number of invalid framing words have been detected. Once frame alignment is accomplished, the frame status block outputs a frame alignment (Falign) pulse that indicates start of frame and is coincident with the M bit. The Hclk is then coincident with the major UIPM overhead bits and the position of all overhead bits (including the ones inside the subframes) is known.

The Falign pulse is used in the timing chain block along with Hclk and f_p to create timing signals for decoding the C bits (Cclk) and for gapping the edges of f_p that correspond to all the UIPM overhead bits (Sclk, 5gap, and Rgap). The Sclk signal is used with the C bit decoder to create a gapping signal (Sgap) corresponding to times when the S bit carries a dummy bit. For the suggested Phase 1 UIPM demultiplexer circuit, the information contained in the O_d and O_r bits are entered manually into the timing chain block providing knowledge about the location of the R bits. All the gapping signals are OR'd with f_p to create a gapped clock that only allows tributary data bits to be read into the buffer. The gapped payload clock is further processed by the PLL to reproduce the tributary clock used to read the data bits out of the buffer.

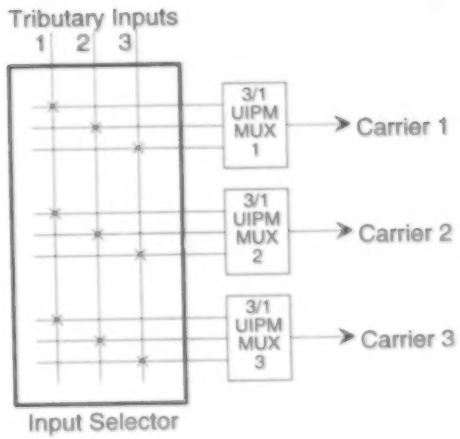
7.3.4 Hardware Considerations for Future UIPM Systems

Subsequent phases of the UIPM concept would eventually accommodate multiple tributaries and multiple carriers resulting in an n/m UIPM system. This requires changes to all aspects of the UIPM system as already discussed. But from the perspective of product development, it also requires development of system diagnostics, built-in systems test (BIST) functions, and various modes of operation.

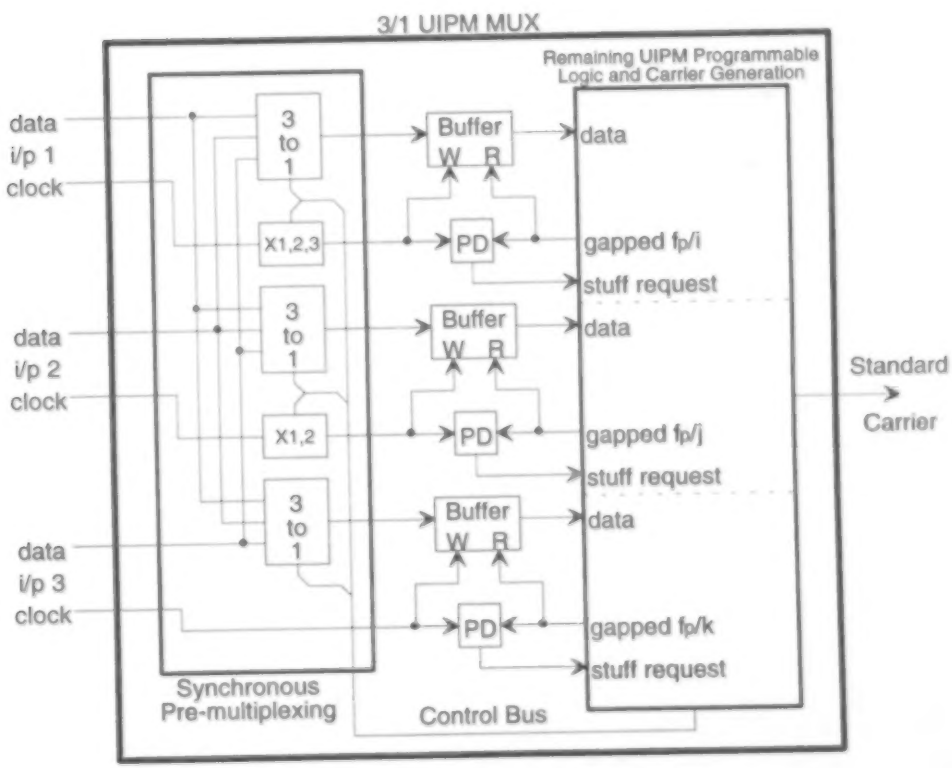
From a practical consideration, implementation of commercial UIPM systems would benefit from major use of VLSI design. Much of the UIPM circuitry could be implemented in VLSI resulting in a small number of UIPM chip sets. Only a small amount of circuitry such as some PLL functions may require use of discrete components.

As mentioned in Section 7.2.3, the most logical way to implement a n/m UIPM system is to have m n/1 UIPM sub-systems where the UIPM format design algorithm determines which of the n tributaries are transported by each of the m carriers. Figure 7.11 shows how this would work for n=m=3 as an example. Figure 7.11a shows three 3/1 UIPM multiplexers (mux) connected to a 3x3 routing switcher. Each UIPM mux can select any of the three inputs. Obviously a given input would be routed to only one UIPM mux. Depending on the rates of the inputs, any combination of inputs to carriers is possible. The optimum combination is determined by the format design algorithm.

Figure 7.11b shows details of each 3/1 UIPM mux. The synchronous pre-multiplexing block allows synchronously related tributaries to be multiplexed into a higher rate tributary before being mapped into a multi-frame UIPM format. Each tributary is then read into a buffer at its original rate and read out by a gapped f_p/x clock where x is determined by the format design algorithm and can be different for each tributary (e.g. x = i, j, or k in Figure 7.11b). In this way, f_p is divided among the inputs, and each input undergoes independent pulse stuffing justification. The gaps in f_p/x correspond to UIPM framing bits, R bits, other overhead bits, and S bits stuffed with dummy bits. The outputs of all buffers are then mapped into the appropriate time slots of a multi-frame UIPM format which becomes the payload of a standard carrier. This process involves interleaving the data bits of all tributaries in some fashion and combining them with the rest of the UIPM format bits. Finally, the standard carrier overhead is added in a manner similar to that described in Section 7.3.2



a) Three 3/1 UIPM multiplexers with selectable inputs



b) Detail of a 3/1 UIPM multiplexer

Figure 7.11 Multiplexer portion of an n/m UIPM system where n=m=3

7.4 SUMMARY

Future development of the UIPM system involves modifications to the related areas software, hardware, and generic format structure. Primary emphasis is given to development of a complete Phase 1 UIPM prototype, but major modifications in all areas would be required for subsequent phases involving multiple tributaries and multiple carriers.

Further development of the UIPM generic format structure includes definition of the O_m, O_d, O_r, and E bits, and possible modifications resulting in improved jitter reduction and increased flexibility enabling the carriage of all possible tributary rates. Further development of the UIPM format design algorithm includes modifications concerning the related issues of efficiency, format size, and jitter performance, as well as manipulation of carrier S bits and possible use of stuff threshold modulation. Finally, further development of the UIPM hardware requires modification and extension of the existing experimental hardware to include data path components and carrier overhead addition and removal circuitry. The UIPM hardware must also reflect and support modifications and enhancements in the UIPM generic format structure and format design algorithm.

8 CONCLUSIONS

As technologically advanced as the telecommunications industry is to-date, it provides no means to transport isochronous signals that do not rigidly conform to its standards. The ability to do so has the potential of increasing the applications base for telecommunication service providers. This is evident by the accelerated research and development of ATM. But ATM is also limited in its ability to provide a near virtual wire for nonstandard isochronous digital signals of arbitrary rate.

This thesis has addressed a possible alternative transport mechanism describing a novel technique called *Universal Isochronous Payload Multiplexing* (UIPM). UIPM employs an algorithm to design a near optimum customized plesiochronous multiplex format for any arbitrary-rate isochronous digital signal. This algorithm is used to configure programmable hardware to map the signal into the payload portion of a standard telecommunication carrier. Proof-of-concept of the UIPM system as well as an original analysis of its performance and contributions to its practical realization has been given.

One unique aspect of UIPM is that no modifications or alterations to the formats of standard telecommunications carriers are necessary. Since only the payload portion of the standard carrier is used, standardization of UIPM is a non issue, and commercial development can occur rapidly.

8.1 SUMMARY OF WORK

The UIPM system consists of three main components: a variable length format structure (called the UIPM generic format), the UIPM format design algorithm, and programmable hardware. Several stages of development of the UIPM system could occur that ultimately result in the transport of multiple nonstandard mixed-rate tributaries over one or more standard carriers. To obtain proof-of-concept, this thesis dealt primarily with the most fundamental (Phase 1) stage of UIPM system development where only one

tributary is mapped into the payload portion of one standard carrier. The carrier chosen was the DS3 signal which offered a large enough payload bandwidth (44.209694 Mb/s) to investigate the performance of UIPM formats designed to map tributaries ranging in rate from 1 Mb/s to about 44 Mb/s.

M [ ] F [ ] C1 [ ] F [ ] C2 [ ] F [ ] C3 [ ] F [S]

[ ] = O_m, O_d, O_s, E, {nD, kR}

[S] = O_m, O_d, O_r, E, S, {(n-1)D, kR}

Figure 8.1 UIPM generic format structure

Figure 8.1 summarizes the UIPM generic format structure. The M bit is for future multi-frame formats and will help to identify start of master frame. The F bits are used for framing and will have a bit pattern such as 1001. The C bits indicate the status of the S bit which is a variable stuff bit used to accommodate the asynchronous relationship between the tributary and the carrier payload rates and to allow for their respective tolerances.

The square brackets indicate subframes which consist of 3 overhead bits (O_m, O_d, O_r), 1 error detection/correction bit (E), and a data element indicated by the curly brackets. The O_m bit is reserved for future use to indicate the number of frames in a multi-frame format. The O_d and O_r bits indicate the sequential pattern of n data (D) and k fixed stuff (R) bits within each element. The variables n and k are determined by the UIPM format design algorithm and gives the generic format its characteristic of being variable in length.

There is 1 stuffing subframe ([S]) per frame which contains the stuff (S) bit. The stuffing subframe contains a data/stuffing element where there are n-1 data bits since the first data bit is replaced by an S bit; all other bits are the same as for a regular subframe.

The UIPM generic format has a minimum and maximum size of 48 and 4104 bits respectively, and a minimum size increment of 8 bits.

The UIPM format design algorithm determines the optimum size, and ratio of D bits to R bits, of the generic format structure for a given tributary rate (f_t) and carrier payload rate (f_p). This is accomplished by finding all format sizes suitable for mapping the tributary into the payload portion of the carrier based on a set of design criteria. The worst case predicted jitter performance (by means of computer simulation) of all suitable format sizes are compared and the one exhibiting the best jitter performance (least waiting time jitter) is selected as optimum for that case.

The experimental hardware used for obtaining proof-of-concept measurements consists of a clock gapper and a PLL. These components represent the main part of the clock processing circuitry required in a complete UIPM system. The clock gapper emulates the pulse stuffing circuitry in the synchronizer portion of a UIPM multiplexer and timing and clock gapping logic necessary in a UIPM demultiplexer. The output of the clock gapper is a gapped payload clock used in a complete UIPM demultiplexer to remove all carrier payload bits that are not tributary data bits. The PLL is used to recover the original tributary clock from the gapped payload clock and represents part of the desynchronizer in a UIPM demultiplexer. This is sufficient hardware to measure actual jitter that would be imparted to the tributary signal due to the mapping process in a real UIPM system.

The UIPM format design algorithm contains a jitter simulation module that models the pulse stuffing synchronizer in a UIPM multiplexer, and the PLL in the desynchronizer portion of a UIPM demultiplexer. Thus, a comparison between predicted and measured jitter results was possible for a select number of tributary test rates that could be accommodated by the experimental hardware. These test rates were a subset of the total range of tributary rates for which worst case predicted jitter was captured.

Worst case predicted jitter was obtained for tributary rates having an offset tolerance of +/-100 ppm ranging from 1 Mb/s to 44 Mb/s in 100 kb/s increments using the DS3 payload rate of 44.209694 Mb/s. Analysis of these results show that in general, the format size tends to increase as the tributary rate approaches the carrier payload rate, resulting in increased jitter. However, over 95% of all rates mentioned above resulted in worst case predicted jitter of less than 0.5 UI p-p. This is a good indicator that transport of nonstandard isochronous digital signals via UIPM technology over the telecommunications network is possible.

Tributary rates whose formats exhibit comparatively high jitter values tend to occur near no-format regions where the UIPM format design algorithm is unable to generate any suitable formats. The regions of no-format rates occur when $f_t \approx f_p(x/y)$ where $x \in \{1,2,3,...,7\}$ and $y \in \{x+1,...,8\}$. Despite the existence of no-format rates, more than 99% of all tributary rates from the minimum possible rate of 75.414 kb/s to the maximum possible rate of 43.77422 Mb/s, in 1 b/s increments (and using the DS3 payload rate), can be accommodated by the UIPM format design algorithm.

Test rates chosen to compare predicted and measured results were: 1, 5, 10, 20, 25.26 (edge of a no-format region), 30, and 36.2 Mb/s (highest rate that could be handled by experimental hardware). These test rates represented a sample of the entire possible range, had format sizes small enough to be handled by the experimental hardware, and provided a reasonable range of predicted p-p jitter values. Relatively good agreement existed between the predicted and measured results which served to verify the UIPM concept and demonstrate the feasibility of its implementation.

Further work on the UIPM system will be necessary for the completion of a Phase 1 prototype as well as subsequent phases involving multiple tributaries and multiple carriers. This will involve further development of the three main UIPM components: the UIPM format design algorithm, UIPM configurable hardware, and the UIPM generic

format structure. These components are inter-related such that development of one may require parallel development of the other two.

8.2 FINAL CONCLUSIONS

This thesis has provided a verification of the UIPM concept and has proven the feasibility of its implementation. It has been demonstrated that a UIPM system could effectively transport nonstandard isochronous digital signals of virtually any rate over the existing telecommunications network without reduction of native rate or loss of inherent synchronization. To the customer desiring transport of an isochronous digital signal, this appears as a virtual wire and the multiplexing procedure is completely transparent. This thesis has also served to identify more complex issues regarding full development of a UIPM system that require further research and study.

Indications are that the back bone of the telecommunications industry will consist largely of Synchronous Transfer Mode signals such as DS3s, STS-Ns, and OC-Ns well into the future, even in the face of emerging technologies such as ATM and BISDN. While ATM continues to evolve and its limitations are realized, UIPM technology can be developed quickly and offers an effective alternative to some of the more questionable claims made of ATM regarding transport of continuous, time-sensitive isochronous digital signals of arbitrary rate. Successful transfer of the UIPM technology from the research phase to the production phase would allow the potential of UIPM to become a reality.

REFERENCES

[Ansi87] American National Standard for Telecommunications, "Digital hierarchy - electrical interfaces," *ANSI Doc. T1.102-1987*, August 1987.

[Ansi88a] American National Standard for Telecommunications, "Digital hierarchy - optical interface rates and formats specifications," *ANSI Doc. T1.105-1988*, September 1988.

[Ansi88b] American National Standard for Telecommunications, "Digital hierarchy - formats specifications," *ANSI Doc. T1.107-1988*, August 1988.

[BaCh89] R. Ballart and Y. Ching, "SONET: Now it's the standard optical network", *IEEE Communications Magazine*, vol. 29, no. 3, pp. 8-15, March 1989.

[Bell86] Bell Communications Research, "Asynchronous digital multiplexer requirements and objectives", *Technical Reference TR-TSY-000009*, Issue 1, May 1986.

[BLab82] Bell Laboratories Staff, *Transmission Systems for Communications*, 5th ed., Western Electric Co. Inc., North Carolina: Winston-Salem, 1982.

[BeBl86] I. Benson, K. Blair, *Television Engineering Handbook*, New York, NY: McGraw Hill Inc., 1986.

[Best84] R.E. Best, *Phase-Locked Loops*, New York, NY: McGraw-Hill Inc., 1984.

[Chow73] P.K. Chow, "Jitter due to pulse stuffing synchronization", *IEEE Trans. Commun.*, vol. COM-21, no. 7, July 1973, pp. 854-859.

[Dutt72] D. Duttweiler, "Waiting time jitter", *Bell System Technical Journal*, vol. 51, pp.165-207, January 1972.

[Gard79] F.M. Gardner, *Phaselock Techniques*, New York, NY: John Wiley & Sons, 1979.

[Grov88] W.D. Grover, "A modified DS3 format and interface circuit for wide area networking at 88 Mb/s over DS3 transport networks", *TR-88-06(R), ATRC (Confidential and Proprietary)*, October 1988.

[Grov89] W.D. Grover, "Technology for transport of arbitrary-rate synchronous customer payloads over the public network", *TR-89-22(R), ATRC (Confidential and Proprietary)*, December 1989.

[GrMo86] W.D. Grover and T.E. Moore, "Effects of fixed stuffing on multiplexer waiting-time jitter", *Proc. of 13th Biennial Symposium on Comm*, pp. A.2.5-A.2.8, June 1986.

[GrMM87] W.D. Grover, T.E. Moore, and J.A. McEachern, "Measured pulse-stuffing jitter in asynchronous DS1/SONET multiplexing with and without stuff-threshold modulation circuit", *Electronics Letters*, vol. 23, no. 18, pp. 959-961, August 1987.

[Lane92] Jim Lane, *Asynchronous Transfer Mode: Bandwidth for the Future*, Telco Systems, Inc., Norwood, MA: Telco Systems, Inc., 1992.

[Lau89] Richard Lau, "Synchronous frequency encoding techniques for B-ISDN circuit emulation", *Society of Photo-optical Instrumentation Engineers*, vol. 1179, pp. 160-171, 1989.

[Lebo92] J.Y. Le Boudec, "The asynchronous transfer mode: a tutorial", *Computer Networks and ISDN Systems*, vol. 24, pp. 279-309, 1992.

[MaKY68] Y. Matsuura, S. Kozuka, and K. Yuki, "Jitter characteristics of pulse stuffing synchronization", *Proc. IEEE Int. Conf. Comm.*, pp. 259-264, June 1968.

[Mino91] D. Minoli, *Telecomunications Techmology Handbook*, Bell Communications Research, Inc., Norwood, MA: Artech House, Inc., 1991.

[Minz89] S.E. Minzer, "Broadband ISDN and asynchronous transfer mode (ATM)", *IEEE Communications Magazine*, vol. 27, no.9, pp. 17-24, September 1989.

[Moor88] T.E. Moore, "Effect of synchronizer stuff threshold crossing detection implementation on waiting time jitter", *Conf. Proc. 14th Biennial Symposium on Communications*, KIngston, Ontario, May, 1988.

[MoKG91] T.E. Moore, R.G. Kusyk and W.D. Grover, "Computer modelling of waiting time jitter in digital transport networks", *Canadian Journal of Electrical and Computer Engineering*, vol. 16, no. 1, pp. 27-33, January 1991.

[Moto89] Motorola Technical Information Center, *MECL device data*, Motorola Inc.,1989.

[OpSc89] A.V. Oppenheim and R.W. Schafer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice Hall Inc., 1989.

[Tele83] Telecom Canada Staff, *Digital Network Notes*, Telecom Canada, Ottawa, Ontario, 1983.

[Say86] Say, S., "DS-3 payload mapping options for SONET", *ECSA T1 contribution T1X1.4/86-462*, November 1986.

[Stal92] W. Stallings, *ISDN And Broadband ISDN*, 2nd ed., New York, NY: Macmillan Publishing House, 1992.

[TrVa89] P.R. Trischitta and E.L. Varma, *Jitter in Digital Transmission Systems*. AT&T Bell laboratories, Norwood, MA: Artech House Inc., 1989.

APPENDIX A

Derivation of Stuff Ratio Equations 1-1 and 1-2

The definition of stuff ratio (ρ) for typical positive pulse stuffing multiplex schemes carrying N plesiochronous tributaries (such as M12 and M23) is the ratio of the average number of stuff opportunities (so) per master frame stuffed with dummy bits to the total number of stuff opportunities available per master frame. Therefore, ρ can be expressed as follows:

$$\rho = \frac{\text{T - oh - average \# time slots used to carry info}}{\text{so}} \tag{A-1}$$

where T is the total number of bits per master frame and oh is the number of overhead bits per master frame. This expression can then be written as:

$$\rho = \frac{\text{T - oh} - \dfrac{N * f_t}{f_f}}{\text{so}}, \quad f_f = \frac{f_s}{T} \tag{A-2}$$

where f_f is the (master) frame rate, f_t is the tributary rate, and f_s is the synchronous carrier rate. The following is also true:

$$T - oh = d + so \tag{A-3}$$

where d is the number of time slots per master frame dedicated for carrying tributary data bits. Equation A-2 can then be written as:

$$\rho = \frac{d + so - \dfrac{N * f_t}{f_f}}{so} \tag{A-4}$$

or expanded more fully as:

$$\rho = \frac{d + so - \dfrac{N * f_t * T}{f_s}}{so} \tag{1-1}$$

For the case where so=1 and N=1 (i.e. one stuff opportunity per frame and only one tributary carried) Equation A-4 then becomes:

$$\rho = d + 1 - \frac{f_t}{f_f} \tag{A-5}$$

but $\frac{f_t}{f_f} = \text{data} + \text{remainder}$, remainder < 1, and remainder $= \text{frac}\left(\frac{f_t}{f_f}\right)$ where frac(x) is the fractional part of x. Substitution of these expressions into A-5 results in:

$$\rho = \mathbf{1} - \mathbf{frac}\left(\frac{\mathbf{f_t}}{\mathbf{f_f}}\right) \tag{1-2}$$

APPENDIX B

Source Code for the UIPM Format Design Algorithm

```
{================================================================
```

UIPM Format Design Algorithm

This program determines optimum UIPM format sizes, and computes the predicted waiting time jitter, for a range of tributary rates adapted into specified carrier payload. Input information is read from a text file. Parameters of the optimum UIPM formats and corresponding predicted jitter results are written out to a text file.

Definition of constants:

numel	number of elements in a frame
oh	number of general dedicated overhead bits (includes all bits except data and fixed stuff bits)
submax	maximum subframe size in bits

Definition of input variables:

fsync	synchronous carrier payload rate (Mb/s)
ppmsync	carrier frequency tolerance in ppm
ppmtrib	tributary frequency tolerance in ppm
ftribmin	minimum tributary rate (Mb/s) in the range of tributary rates to be adapted into the carrier payload
ftribmax	maximum tributary rate (Mb/s) in the range of tributary rates to be adapted into the carrier payload
ftribinc	tributary rate increment (b/s) for range of tributary rates
fsweepinc	frequency increment (Hz) of frequency offset range each tributary is swept over to determine largest p-p jitter for a given feasible format
thresh	phase threshold (UI) between the tributary and carrier payload clocks that initiates a stuff request
fc	desired cutoff frequency (Hz) of the PLL transfer characteristic
damp	desired damping coefficient of the PLL response
samdur	duration in seconds of real time the jitter module uses to accumulate end-of-frame phase samples for computing p-p and rms jitter
filename	name of output file results are written to

Definition of output variables

ftrib	current tributary rate
level	level of optimum UIPM format size
total	total number of bits in an optimum UIPM format
datbits	total number of data bits in an optimum UIPM format

fixbits	total number of fixed stuff bits in an optimum UIPM format
ratio	numerical stuff ratio of an optimum UIPM format
wbrmsmin	predicted wideband rms jitter (UI) associated with the worst case predicted narrowband p-p jitter value, occurring within a tributary's effective frequency offset range, of an optimum UIPM format
rmsmin	predicted narrowband rms jitter (UI) associated with the worst case predicted narrowband p-p jitter value, occurring within a tributary's effective frequency offset range, of an optimum UIPM format
wbppmin	predicted wideband p-p jitter (UI) associated with the worst case predicted narrowband p-p jitter value, occurring within a tributary's effective frequency offset range, of an optimum UIPM format
ppmin	worst case predicted narrowband p-p jitter (UI), occurring within a tributary's effective frequency offset range, for an optimum UIPM format - this is the lowest of all worst case predicted narrowband p-p jitter values among all feasible UIPM format sizes for a given tributary rate

Definition of other local variables:

data	current number of data bits per frame
eldata	current number of data bits per data element
elmax	maximum number of bits in a data element
eltot	current number of total bits in a data element
fixed	current number of fixed stuff bits per frame
fsweep	current tributary rate existing within its effective frequency offset range
fsweepmax	maximum effective tributary rate that can be accommodated by a feasible UIPM format size
fsweepmin	minimum effective tributary rate that can be accommodated by a feasible UIPM format size
gfin	input text file used to initialize input variables
gfout	output text file of name "filename" that results are written to
lev	current level of format size
numsam	number of samples corresponding to the duration in seconds of real time the jitter module uses to accumulate end-of-frame phase samples for computing p-p and rms jitter of a feasible format size
pp	predicted narrowband p-p jitter (UI) for the current level
ppmax	worst case predicted narrowband p-p jitter (UI), occurring within a tributary's effective frequency offset range, for a feasible UIPM format size
rms	predicted narrowband rms jitter (UI) for the current level
rmsmax	predicted narrowband rms jitter (UI) associated with the worst case predicted narrowband p-p jitter value, occurring within a tributary's effective frequency offset range, of a feasible UIPM format size
sr	numerical stuff ratio of a feasible UIPM format size
tot	total number of bits per frame for the current level
totmax	maximum number of total bits per frame
wbpp	predicted wideband p-p jitter (UI) for the current level

```
 wbppmax     predicted wideband p-p jitter (UI) associated with the worst case predicted
             narrowband p-p jitter value, occurring within a tributary's effective
             frequency offset range, of a feasible UIPM format size
 wbrms       predicted wideband rms jitter (UI) for the current level
 wbrmsmax    predicted wideband rms jitter (UI) associated with the worst case predicted
             narrowband p-p jitter value, occurring within a tributary's effective
             frequency offset range, of a feasible UIPM format size
=============================================================================}

program GenericFormat7;

const
    numel     = 8;
    oh        = 40;
    submax    = 512;

var
    data,datbits,elmax,eltot,fixed,fixbits,
    lev,level,numsam,numtrans,tot,total,totmax                : integer32;

    damp,eldata,fc,fr,fsync,
    ftrib,ftribinc,ftribmax,ftribmin,
    fsweep,fsweepinc,fsweepmax,fsweepmin,
    ppmtrib,ppmsync,pp,ppmax,ppmin,ratio,
    rms,rmsmax,rmsmin,samdur,sr,thresh,
    wbrms,wbrmsmax,wbrmsmin,wbpp,wbppmax,wbppmin : real;

    filename                                                  : string;

    gfin,gfout                                                : text;
{
gfout cols: 1=ftrib, 2=level, 3=total, 4=datbits, 5=fixbits, 6=ratio, 7=wbrmsmin,
8=rmsmin, 9=wbppmin, 10=ppmin.
}
procedure getjitter(ftrib,fsync,fr,fc,damp,thresh: real;
                    numsam,data,fixed,oh: integer32;
                    var wbrms,rms,wbpp,pp: real);
external;

function frac(x: real): real;
    begin
        frac:=x-trunc(x);
    end;
```

```
begin {main prog}

    totmax:=8+submax*8;

    open(gfin,'gfin','old');
    reset(gfin);

    {read in synchronous output rate (Mb/s)}
    readln(gfin,fsync);
    fsync:=fsync*1E6;
    {read in synchronous carrier tolreance (ppm)}
    readln(gfin,ppmsync);
    ppmsync:=ppmsync*1E-6;

    {read in tributary tolerance (ppm)}
    readln(gfin,ppmtrib);
    ppmtrib:=ppmtrib*1E-6;

    {read in ftribmin and ftribmax (Mb/s)}
    readln(gfin,ftribmin);
    ftribmin:=ftribmin*1E6;
    readln(gfin,ftribmax);
    ftribmax:=ftribmax*1E6;
    {read in tributary frequency increment (Hz)}
    readln(gfin,ftribinc);
    {read in frequency sweep increment (Hz)}
    readln(gfin,fsweepinc);

    {read in desired threshold (UI)}
    readln(gfin,thresh);
    {read in cutoff freq for desync PLL model (Hz)}
    readln(gfin,fc);
    {read in desired damping coefficient of PLL model}
    readln(gfin,damp);
    {read in sample duration (sec)}
    readln(gfin,samdur);

    {read in name of output file}
    readln(gfin,filename);

    close(gfin);

    open(gfout,filename,'unknown');
    rewrite(gfout);
```

```
elmax:=(totmax-oh) div numel;
ftrib:=ftribmin;

{find UIPM formats for all ftribmin≤ftrib≤ftribmax in increments of ftribinc}
while ftrib<=ftribmax do
    begin  {main kernel}
        eltot:=0;
        lev:=0;
        ppmin:=1000;
        fsweepmin:=ftrib-ftrib*(ppmtrib+ppmsync*fsync/ftrib);
        fsweepmax:=ftrib+ftrib*(ppmtrib+ppmsync*fsync/ftrib);
        writeln;
        repeat
            lev:=lev+1;

            repeat {find a feasible format for the current level}
                eltot:=eltot+1;
                tot:=oh+eltot*numel;
                fr:=fsync/tot;
                data:=trunc(ftrib/fr+1E-6);  {...+1E-6 ensures correct integer value}
                if (frac(ftrib/fr+1E-6)<=1E-6) and (frac(data/numel)=0) then
                    data:=data-1;
                eldata:=(data+1)/numel;
            until (eltot=elmax) or ( (frac(eldata)=0) and  {←ensures integral number of
                                                            data bits in element}
                                     (eltot>=eldata) and  {←ensures total bits per element
                                                            is ≥ data bits per element}
                                     (eldata>=1) and  {←ensures there is at least one data
                                                        bita per element}
                                     (data*fr<=fsweepmin) and      {←ensures adequate
                                                                    stuffing range}
                                     ((data+1)*fr>=fsweepmax) );  {←ensures adequate
                                                                    stuffing range}

            if   (frac(eldata)=0) and
                 (eltot>=eldata) and
                 (eldata>=1) and
                 (data*fr<=fsweepmin) and
                 ((data+1)*fr>=fsweepmax) then
                 begin  {compute predicted worst case jitter for current format size}
                     fixed:=tot-data-1-oh;
                     sr:=1-frac(ftrib/fr+1E-6);
                     if (frac(ftrib/fr+1E-6)<=1E-6) and(frac(data/numel)=0) then
                         sr:=0;
```

```
{determine number of phase samples to accumulate based on real
time sample duration}
numsam:=round(fr*samdur);

ppmax:=0;
fsweep:=fsweepmin;
while fsweep<=fsweepmax do
    begin {call jitter simulation module for each value of fsweep}
        getjitter(fsweep,fsync,fr,fc,damp,thresh,
                numsam,data,fixed,oh,
                wbrms,rms,wbpp,pp);
        if pp>ppmax then {store results associated with worst case
                          p-p jitter}
            begin
                wbrmsmax:=wbrms;
                rmsmax:=rms;
                wbppmax:=wbpp;
                ppmax:=pp; {←worst case p-p jitter for current
                            format size}
            end;
        fsweep:=fsweep+fsweepinc;
    end;

{store format parameters and predicted jitter results of format with
lowest worst case predicted p-p jitter}
if ppmax<ppmin then
    begin
        total:=tot;
        datbits:=data;
        fixbits:=fixed;
        ratio:=sr;
        wbrmsmin:=wbrmsmax;
        rmsmin:=rmsmax;
        wbppmin:=wbppmax;
        ppmin:=ppmax;
        level:=lev;
    end;

end

{if no feasible formats could be found, store dummy values}
else if lev=1 then
    begin
        total:=0;
        datbits:=0;
```

```
                    fixbits:=0;
                    level:=0;
                    ratio:=-0.1;
                    wbrmsmin:=-0.05;
                    rmsmin:=-0.05;
                    wbppmin:=-0.05;
                    ppmin:=-0.05;
                    level:=1;
                end;

            until eltot=elmax;  {UIPM generic format reaches its maximum size}

            {write results to output file}
            writeln(gfout,ftrib:10:6,' ',level,' ',total,' ',datbits,' ',
                fixbits,' ',ratio:8:6,' ',wbrmsmin:8:6,' ',
                rmsmin:8:6,' ',wbppmin:8:6,' ',ppmin:8:6,chr(13));

            ftrib:=ftrib+ftribinc;  {find optimum UIPM format for next tributary rate}

        end;  {main kernel}

    close(gfout);

end.  {main program}
```

{==========

Jitter Simulation Module

This procedure uses variables determined by the main UIPM format design program to compute wideband and narrowband p-p and rms waiting time jitter of a feasible UIPM format size.

Definition of local variables:

a0,a1,a2	coefficients in the numerator of the transfer function for an IIR discrete-time filter
b0,b2,b2	coefficients in the denominator of the transfer function for an IIR discrete-time filter
deltaphi	relative change in phase between the tributary and carrier payload clocks during the carrier payload clock period when a time slot contains a data bit
fil	filtered frame-end phase samples from the IIR discrete-time filter
fil1.fil2	delayed frame-end outputs of the IIR filter
filmax	largest filtered phase sample over the duration of the sampling period
filmin	smallest filtered phase sample over the duration of the sampling period
filsum	sum of all filtered phase samples over the duration of the sampling period
filsumsq	square of filsum
fn	natural frequency (Hz) of a second order system
i	an integer counter
numtrans	number of transient phase samples allowed to occur before valid samples are accumulated - this corresponds to the settling time of the discrete-time filter used to model the PLL characteristic
phi	frame-end phase difference between the tributary and carrier payload clocks
phi1,phi2	delayed frame-end phase samples used in the IIR filter implementation
phimax	largest frame-end phase sample over the duration of the sampling period
phimin	smallest frame-end phase sample over the duration of the sampling period
phisum	sum of all phase samples over the duration of the sampling period
phisumsq	square of phisum
warp	pre-warping expression

==========}

```
module jitter;

procedure getjitter(ftrib,fsync,fr,fc,damp,thresh: real;
                    numsam,data,fixed,oh: integer32;
                    var wbrms,rms,wbpp,pp: real);
const
    pi = 3.1415926535897932385;

var
    numtrans,i                                          : integer32;

    deltaphi,fn,warp,b0,b1,b2,a0,a1,a2,
    phi,phi1,phi2,phisum,phisumsq,phimax,phimin,
    fil,fil1,fil2,filsum,filsumsq,filmax,filmin          : real;

begin

    {determine fn based on type of PLL characteristic modeled}
    fn:=fc/sqrt(1-2*sqr(damp)+sqrt(1+sqr(2*sqr(damp)-1)));

    deltaphi:=fsync/ftrib-1; {in read clock UI}

    {calculate IIR filter parameters based on type of PLL characteristic modeled}
    warp:=sin(pi*fn/fr)/cos(pi*fn/fr);
    b0:=1+2*damp*warp+sqr(warp);
    b1:=2*(1-sqr(warp))/b0;
    b2:=-(1-2*damp*warp+sqr(warp))/b0;
    a0:=warp*(2*damp+warp)/b0;
    a1:=2*sqr(warp)/b0;
    a2:=warp*(warp-2*damp)/b0;

    {initialize phase and filter variables}
    phi:=thresh;
    phi1:=thresh;
    phi2:=thresh;
    phisum:=0;
    phisumsq:=0;
    phimax:=0;
    phimin:=1000;
    fil1:=thresh;
    fil2:=thresh;
    filsum:=0;
    filsumsq:=0;
    filmax:=0;
    filmin:=1000;
```

```
{allow a transient period equivalent to 20 time constants to occur before phase
samples are accumulated; this allows the filter output to reach steady state}
numtrans:=20*round(fr/damp/2/pi/fn);

for i:=1 to numtrans do
    begin
        if phi>thresh then
            phi:=phi+data*deltaphi-fixed-oh-1
        else
            phi:=phi+(data+1)*deltaphi-fixed-oh;
        fil:=a0*phi+a1*phi1+a2*phi2+b1*fil1+b2*fil2;
        phi2:=phi1;
        phi1:=phi;
        fil2:=fil1;
        fil1:=fil;
    end;

{accumulate unfiltered and filtered phase samples for the duration of the sampling
period}
for i:=1 to numsam do
    begin

        if phi>thresh then
            phi:=phi+data*deltaphi-fixed-oh-1
        else
            phi:=phi+(data+1)*deltaphi-fixed-oh;
        fil:=a0*phi+a1*phi1+a2*phi2+b1*fil1+b2*fil2;

        phi2:=phi1;
        phi1:=phi;
        fil2:=fil1;
        fil1:=fil;

        phisum:=phisum+phi;
        phisumsq:=phisumsq+sqr(phi);
        if phimax<phi then
            phimax:=phi;
        if phimin>phi then
            phimin:=phi;

        filsum:=filsum+fil;
        filsumsq:=filsumsq+sqr(fil);
        if filmax<fil then
            filmax:=fil;
```

```
            if filmin>fil then
                filmin:=fil;

        end;

    {determine wideband rms jitter by calculating the standard deviation of the
    accumulated unfiltered phase samples}
    if phisumsq/numsam<sqr(phisum/numsam) then {to prevent sqrt(-N)}
        wbrms:=0
    else
        wbrms:=sqrt(phisumsq/numsam-sqr(phisum/numsam))*ftrib/fsync;

    {determine wideband p-p jitter}
    wbpp:=(phimax-phimin)*ftrib/fsync;

    {determine narrowband rms jitter by calculating the standard deviation of the
    accumulated filtered phase samples}
    if filsumsq/numsam<sqr(filsum/numsam) then {to prevent sqrt(-N)}
        rms:=0
    else
        rms:=sqrt(filsumsq/numsam-sqr(filsum/numsam))*ftrib/fsync;

    {determine narrowband p-p jitter}
    pp:=(filmax-filmin)*ftrib/fsync;

    {ftrib/fsync converts values to write clock UI}

end; {jitter module}
```

APPENDIX C

Derivation of Cutoff Frequency Expressions (Eqns. 4-8 and 4-9)

This appendix derives the expressions for the cutoff frequency of the two PLL transfer functions discussed in Chapter 4.

Derivation of Equation 4-8

The PLL transfer function for the Butterworth-type second order system is given by:

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4-6}$$

The cutoff frequency (ω_c) is the frequency at which the power amplitude is one half (or -3 dB) the power amplitude at 0 frequency. Therefore, ω_c can be solved for by squaring the absolute value of $H(s=j\omega_c)$ and equating it to 1/2 as shown in the following steps:

$$|H(j\omega_c)|^2 = \left|\frac{\omega_n^2}{(j\omega_c)^2 + 2\zeta j\omega_c\omega_n + \omega_n^2}\right|^2 = \frac{1}{2} \tag{C-1}$$

$$= \frac{\omega_n^4}{\left|\omega_n^2 - \omega_c^2 + j2\zeta\omega_c\omega_n\right|^2}$$

$$= \frac{\omega_n^4}{(\omega_n^2 - \omega_c^2)^2 + (2\zeta\omega_c\omega_n)^2}$$

$$\therefore (\omega_n^2 - \omega_c^2)^2 + (2\zeta\omega_c\omega_n)^2 = 2\omega_n^4 \tag{C-2}$$

Expansion and grouping of like terms results in:

$$\omega_c^4 + \omega_c^2[2\omega_n^2(2\zeta^2 - 1)] - \omega_n^4 = 0 \tag{C-3}$$

or equivalently

$$ax^2 + bx + c = 0 \tag{C-4}$$

where a=1, $b = 2\omega_n^2(2\zeta^2 - 1)$, $c = -\omega_n^4$, and $x = \omega_c^2$. This is a quadratic equation that can be solved by the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{C-5}$$

The solution for ω_c^2 is then given by:

$$\omega_c^2 = \frac{-2\omega_n^2(2\zeta^2 - 1) \pm \sqrt{[2\omega_n^2(2\zeta^2 - 1)]^2 + 4\omega_n^4}}{2}, \tag{C-6}$$

and the final solution for ω_c is:

$$\omega_c = \omega_n \sqrt{1 - 2\zeta^2 + \sqrt{(2\zeta^2 - 1)^2 + 1}}. \tag{4-8}$$

Derivation of Equation 4-9

The starting point for the derivation of Equation 4-9 is the PLL transfer function given by:

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4-7}$$

The rest of the derivation follows the same procedure as for Equation 4-8:

$$|H(j\omega_c)|^2 = \left| \frac{2\zeta j\omega_c\omega_n + \omega_n^2}{(j\omega_c)^2 + 2\zeta j\omega_c\omega_n + \omega_n^2} \right|^2 = \frac{1}{2} \tag{C-7}$$

$$= \left| \frac{\omega_n^2 + j2\zeta\omega_c\omega_n}{\omega_n^2 - \omega_c^2 + j2\zeta\omega_c\omega_n} \right|^2$$

$$= \frac{\omega_n^4 + (2\zeta\omega_c\omega_n)^2}{(\omega_n^2 - \omega_c^2)^2 + (2\zeta\omega_c\omega_n)^2}$$

$$\therefore (\omega_n^2 - \omega_c^2)^2 + 4(\zeta\omega_c\omega_n)^2 = 2\omega_n^4 + 8(\zeta\omega_c\omega_n)^2 \tag{C-8}$$

Expansion and grouping of like terms results in:

$$\omega_c^4 - \omega_c^2[2\omega_n^2(2\zeta^2 + 1)] - \omega_n^4 = 0 \tag{C-9}$$

or equivalently

$$ax^2 + bx + c = 0 \tag{C-10}$$

where a=1, $b = -2\omega_n^2(2\zeta^2 + 1)$, $c = -\omega_n^4$, and $x = \omega_c^2$. The solution for ω_c^2 is again found by using the quadratic formula:

$$\omega_c^2 = \frac{2\omega_n^2(2\zeta^2 + 1) \pm \sqrt{[2\omega_n^2(2\zeta^2 + 1)]^2 + 4\omega_n^4}}{2}, \tag{C-11}$$

and the final solution for ω_c is:

$$\omega_c = \omega_n \sqrt{1 + 2\zeta^2 + \sqrt{(2\zeta^2 + 1)^2 + 1}}. \tag{4-9}$$

APPENDIX D

Second Order IIR LPF Design and Implementation

This Appendix derives the expressions used in the jitter simulation module of the UIPM format design algorithm to implement an IIR LPF. This discrete-time filter is used to model a second order PLL transfer function. Two derivations are given: one for the transfer function expressed in Equation 4-6 (Butterworth-type response), and one for the transfer function expressed in Equation 4-7 (models actual hardware PLL).

IIR LPF Design for Equation 4-6

The transfer function given by Equation 4-6 is:

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4-6}$$

Conversion from the analog transfer function H(s) to the discrete-time transfer function H(z) is accomplished with the bilinear transform:

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right), \; T = \frac{1}{f_f} \tag{D-1}$$

where T is the sampling period and is the reciprocal of f_f, the frame rate of a given UIPM format design. Use of the bilinear transform results in distortion or warping of the frequency axis when frequencies from the s-plane are mapped into the z-plane. Consequently, the analog frequencies are "pre-warped" before conversion to preserve the analog frequency response. The pre-warping expression is:

$$\omega' = \frac{2}{T}\tan\left(\frac{\omega T}{2}\right) = 2f_f \tan\left(\frac{\pi f}{f_f}\right) \tag{D-2}$$

where ω' is the pre-warped frequency and $\omega = 2\pi f$.

The pre-warped analog transfer function is then:

$$H(s)_{pre\text{-}warped} = \frac{\omega'^{2}_{n}}{s^2 + 2\zeta\omega'_{n}\, s + \omega'^{2}_{n}} \tag{D-3}$$

Application of the bilinear transform results in:

$$H(z) = \frac{\omega'^{2}_{n}}{\left[2f_f\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]^2 + 2\zeta\left[2f_f\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]\omega'_{n} + \omega'^{2}_{n}} \tag{D-4}$$

Multiplying the numerator and denominator by $\frac{\left(1+z^{-1}\right)^2}{4f_f^2}$ and grouping like terms results in:

$$H(z) = \frac{\left(\frac{\omega'_n}{2f_f}\right)^2\left(1+2z^{-1}+z^{-2}\right)}{\left[1+2\zeta\left(\frac{\omega'_n}{2f_f}\right)+\left(\frac{\omega'_n}{2f_f}\right)^2\right]-2\left[1-\left(\frac{\omega'_n}{2f_f}\right)^2\right]z^{-1}-\left[-1+2\zeta\left(\frac{\omega'_n}{2f_f}\right)-\left(\frac{\omega'_n}{2f_f}\right)^2\right]z^{-2}} \tag{D-5}$$

In this form, Equation D-5 can be equated to Equation 4-10:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} - b_2 z^{-2}} \tag{4-10}$$

Let $\frac{\omega'_n}{2f_f} = warp$, then:

$$warp = \tan\left(\frac{\pi f_n}{f_f}\right) \tag{D-6}$$

$$b_0 = 1 + 2\zeta warp + warp^2 \tag{D-7}$$

$$b_1 = \frac{2\left(1 - warp^2\right)}{b_0} \tag{D-8}$$

$$b_2 = \frac{-\left(1 - 2\zeta warp + warp^2\right)}{b_0} \tag{D-9}$$

$$a_0 = \frac{warp^2}{b_0} \tag{D-10}$$

$$a_1 = 2a_0 \tag{D-11}$$

$$a_2 = a_0 \tag{D-12}$$

Equations D-7 to D-12 represent the coefficients of Equation 4-10 used to model the PLL transfer function given by Equation 4-6.

IIR LPF Design for Equation 4-7

The transfer function given by Equation 4-7 is:

$$H(s) = \frac{\Phi out(s)}{\Phi in(s)} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4-7}$$

After pre-warping the analog frequencies and applying the bilinear transform the discrete-time transfer function H(z) becomes:

$$H(z) = \frac{2\zeta\left[2f_f\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]\omega'_n + \omega'^2_n}{\left[2f_f\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]^2 + 2\zeta\left[2f_f\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right]\omega'_n + \omega'^2_n} \tag{D-13}$$

Multiplying the numerator and denominator by $\frac{\left(1+z^{-1}\right)^2}{4f_f^2}$ and grouping like terms results in:

$$H(z) = \frac{\left[2\zeta\left(\frac{\omega'_n}{2f_f}\right) + \left(\frac{\omega'_n}{2f_f}\right)^2\right] + \left[2\left(\frac{\omega'_n}{2f_f}\right)^2\right]z^{-1} + \left[\left(\frac{\omega'_n}{2f_f}\right)^2 - 2\zeta\left(\frac{\omega'_n}{2f_f}\right)\right]z^{-2}}{\left[1 + 2\zeta\left(\frac{\omega'_n}{2f_f}\right) + \left(\frac{\omega'_n}{2f_f}\right)^2\right] - 2\left[1 - \left(\frac{\omega'_n}{2f_f}\right)^2\right]z^{-1} - \left[-1 + 2\zeta\left(\frac{\omega'_n}{2f_f}\right) - \left(\frac{\omega'_n}{2f_f}\right)^2\right]z^{-2}} \tag{D-14}$$

Equating Equation D-14 to Equation 4-10 and using the same definition for *warp* results in the following expressions for the coefficients of Equation 4-10 used to model the PLL transfer function given by Equation 4-7:

$$warp = \tan\left(\frac{\pi f_n}{f_f}\right) \tag{D-15}$$

$$b_0 = 1 + 2\zeta warp + warp^2 \tag{D-16}$$

$$b_1 = \frac{2\left(1 - warp^2\right)}{b_0} \tag{D-17}$$

$$b_2 = \frac{-\left(1 - 2\zeta warp + warp^2\right)}{b_0} \tag{D-18}$$

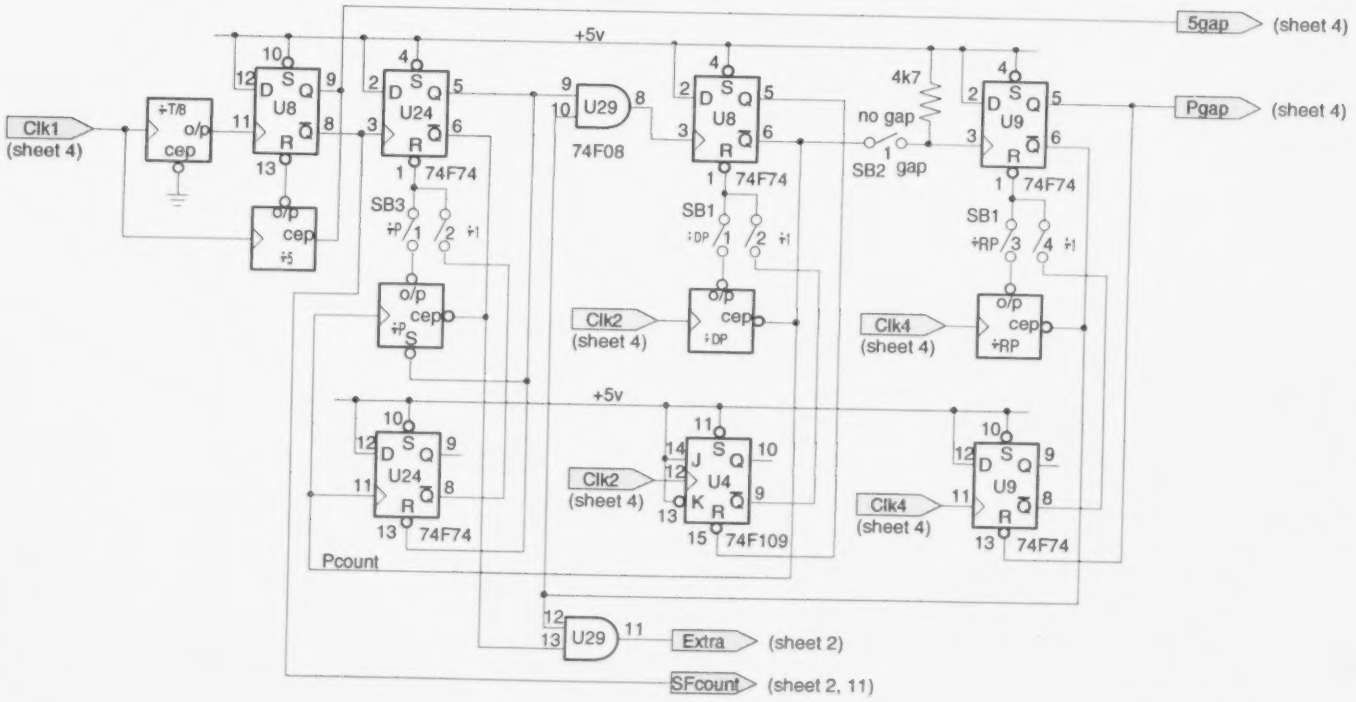
$$a_0 = \frac{warp\left(2\zeta + warp\right)}{b_0} \tag{D-19}$$

$$a_1 = \frac{2warp^2}{b_0} \tag{D-20}$$

$$a_2 = \frac{warp\left(warp - 2\zeta\right)}{b_0} \tag{D-21}$$

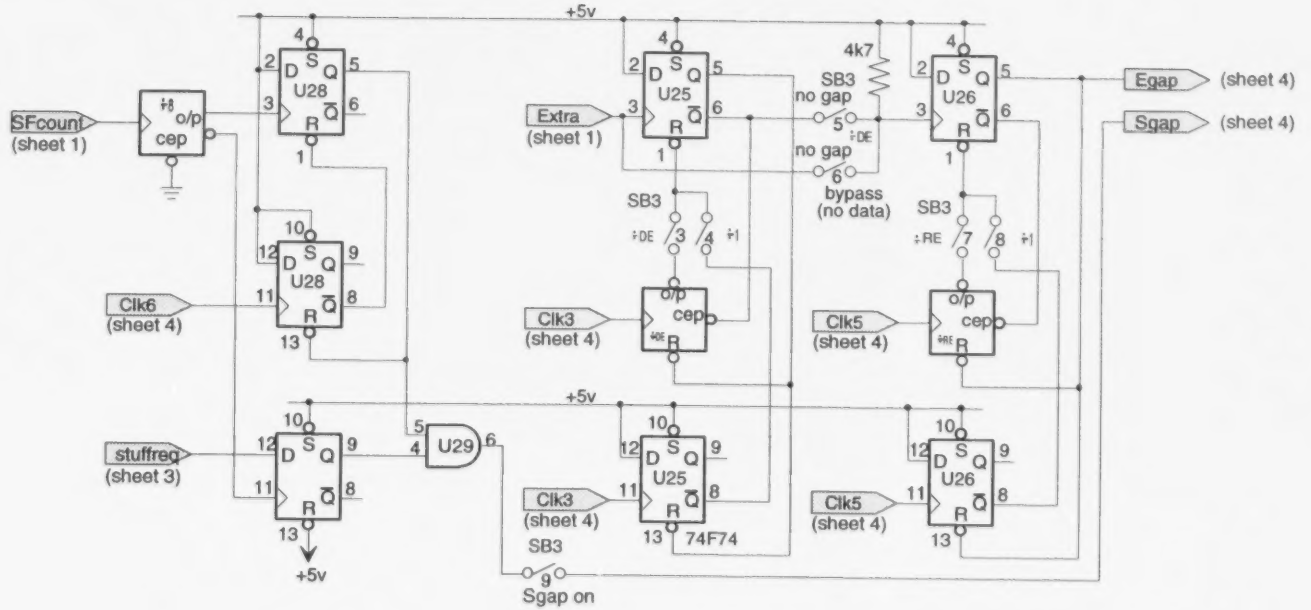
APPENDIX E

Schematics



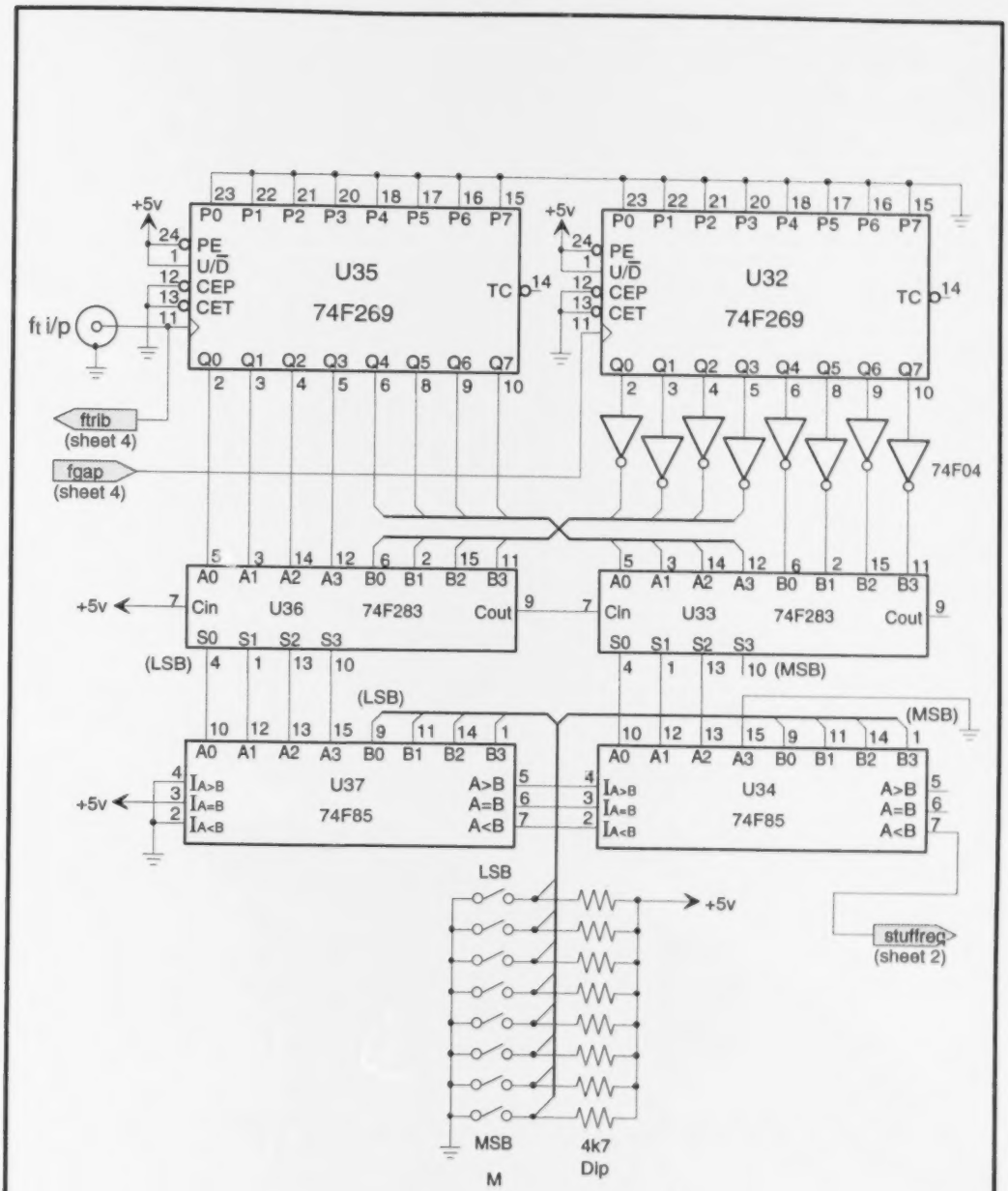
NOTE: SB = Dip Switch Block

Designed by: Tim Friesen	Drawn by: Tim Friesen	Module: Clock Gapper	Circuit: 5gap and Pgap	Sheet #: 1/11	Date: 05/23/94

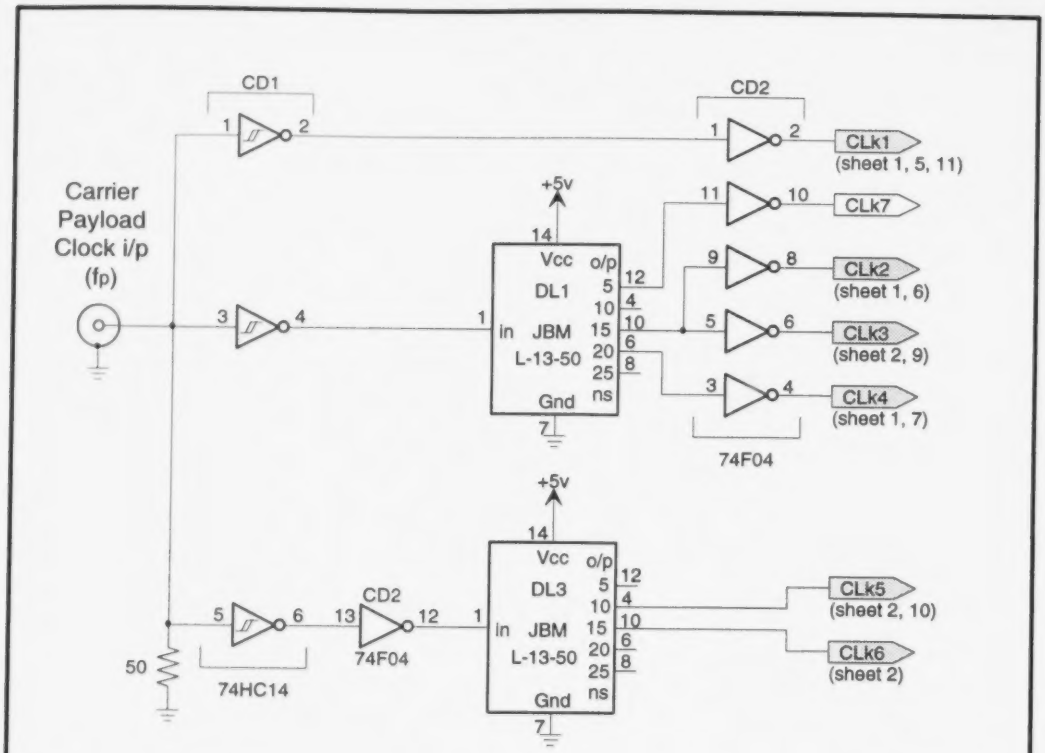


NOTE: SB = Dip Switch Block

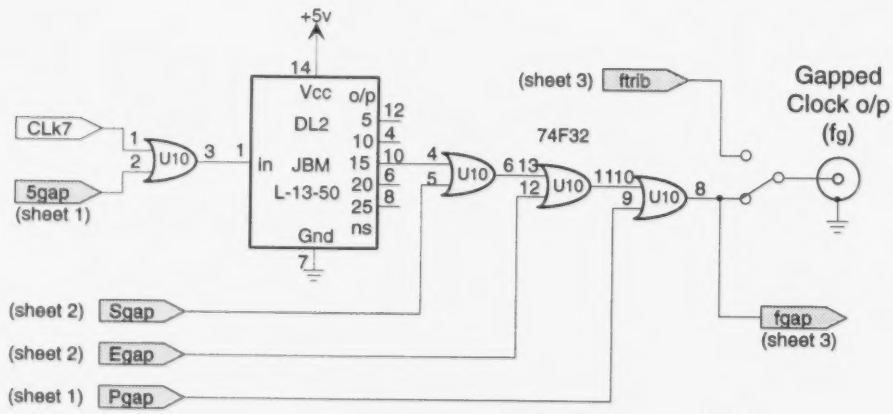
Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	Egap and Sgap	2/11	05/23/94



Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	Variable Stuffing Mechanism	3/11	05/12/94

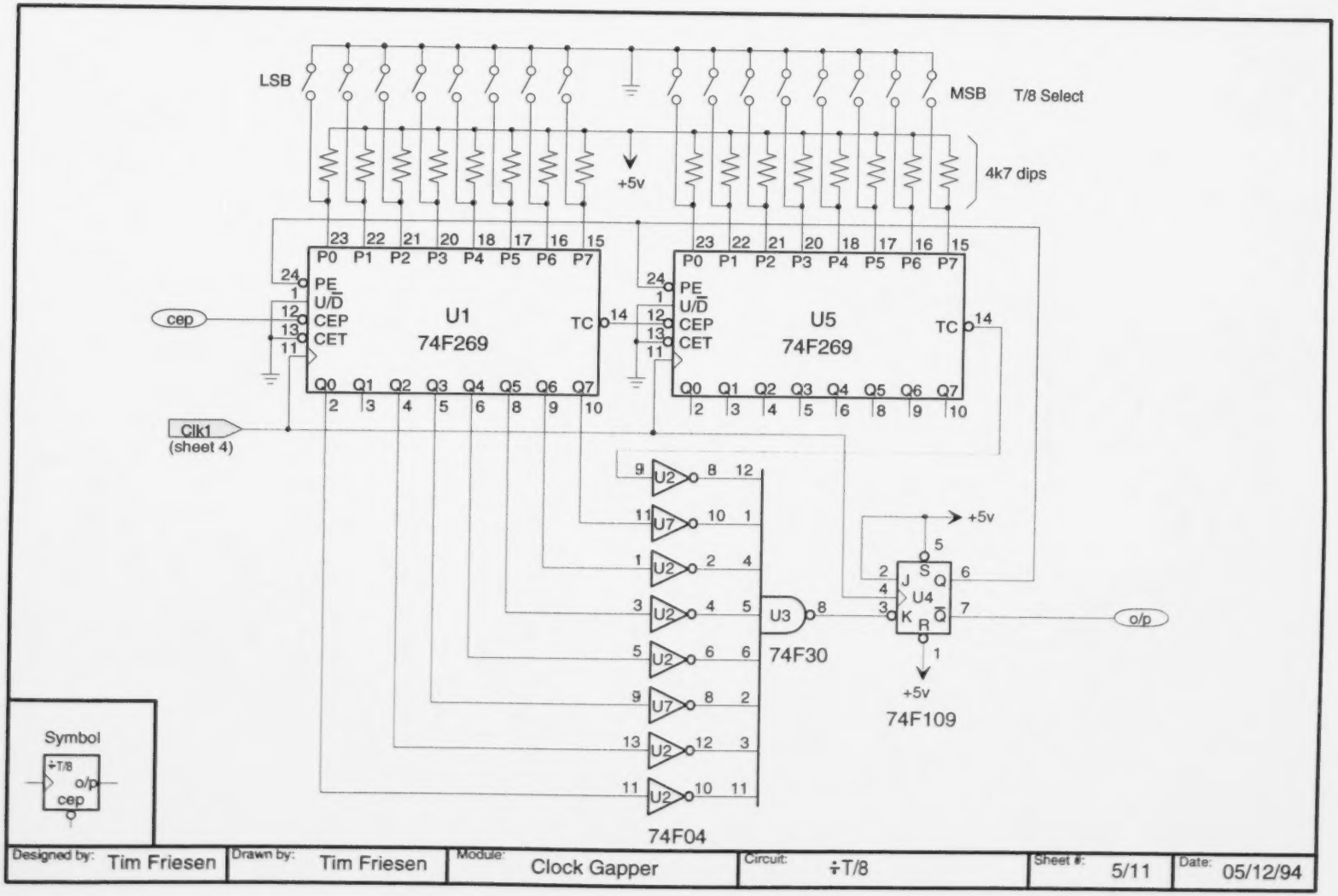


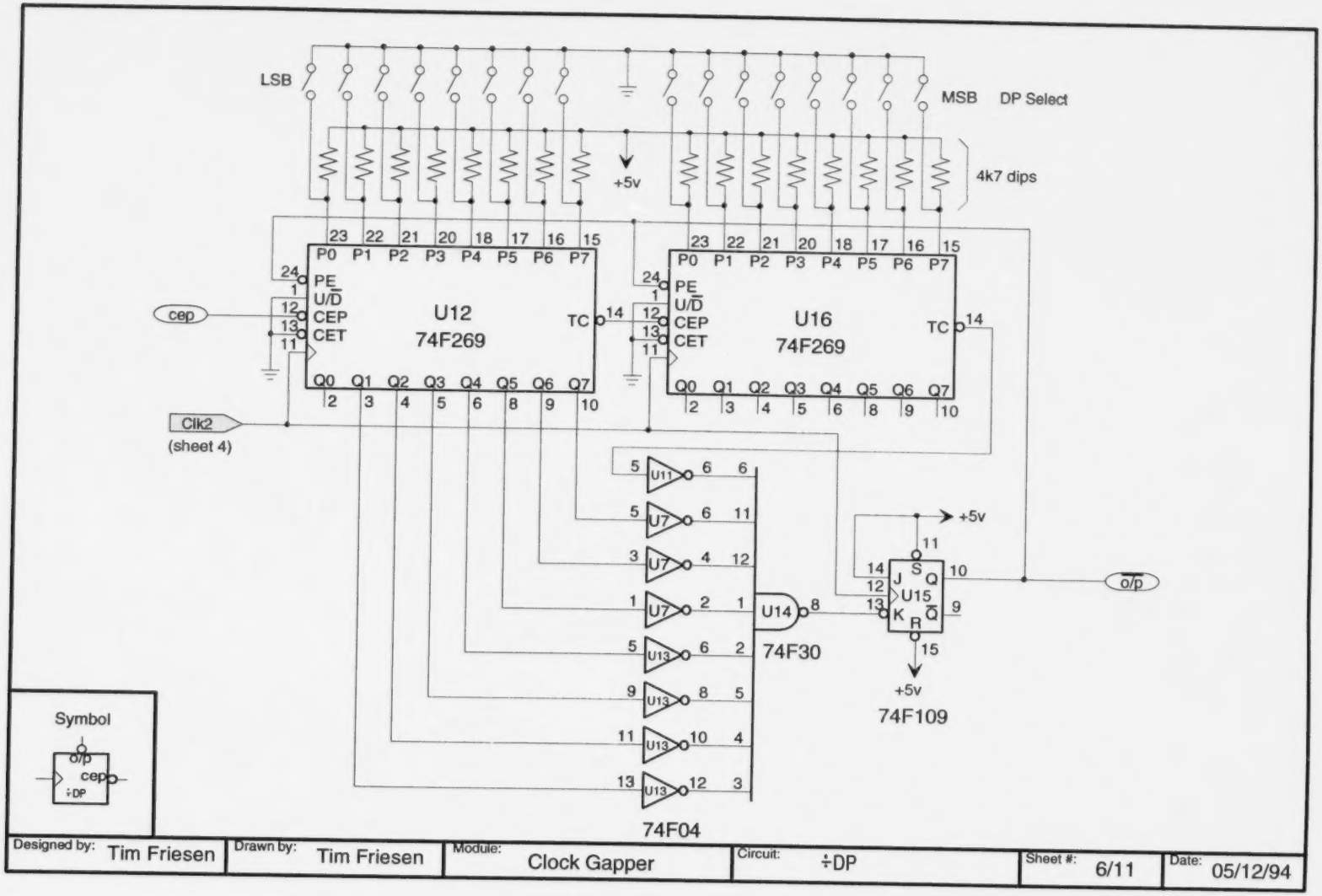
Clock Distibution

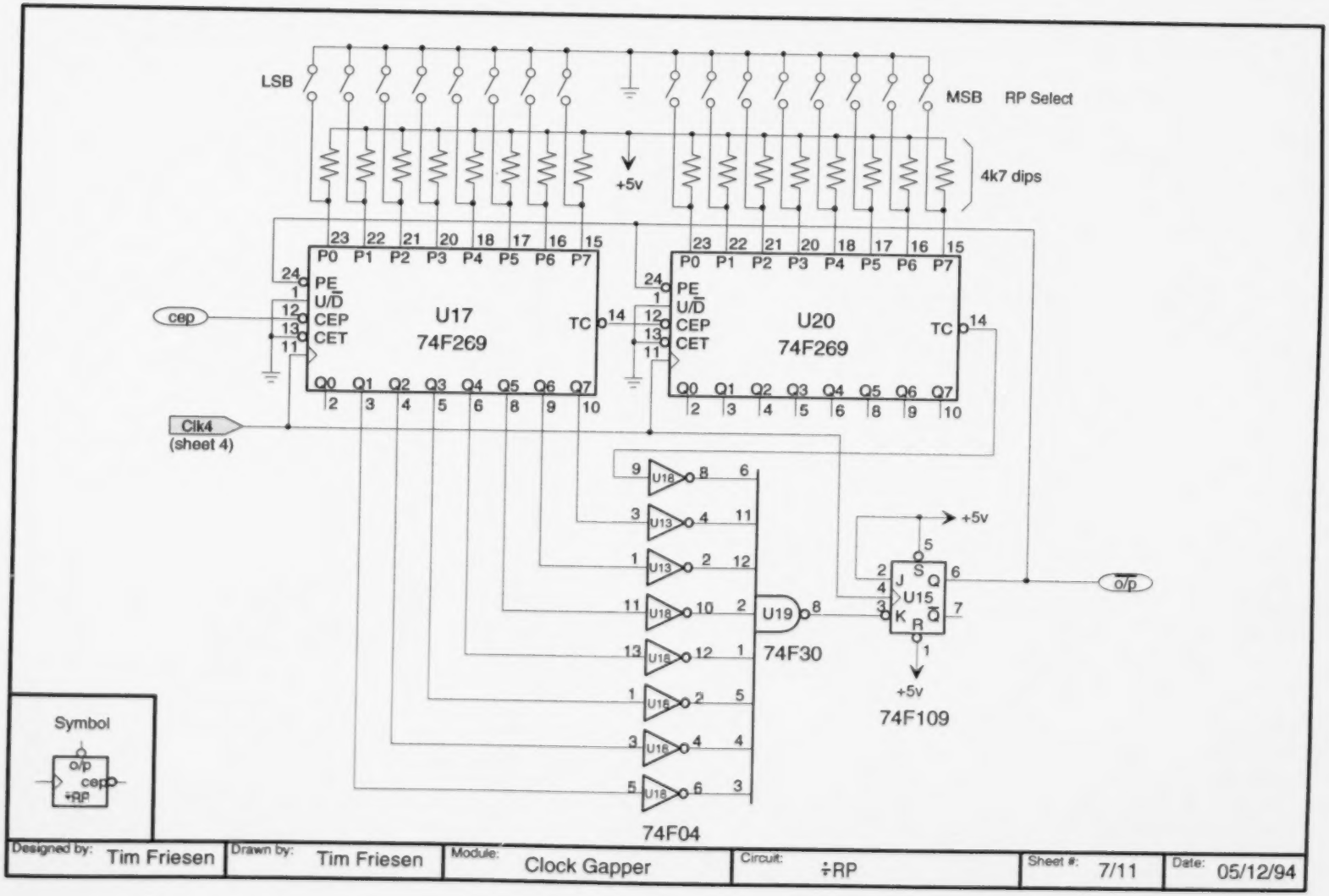


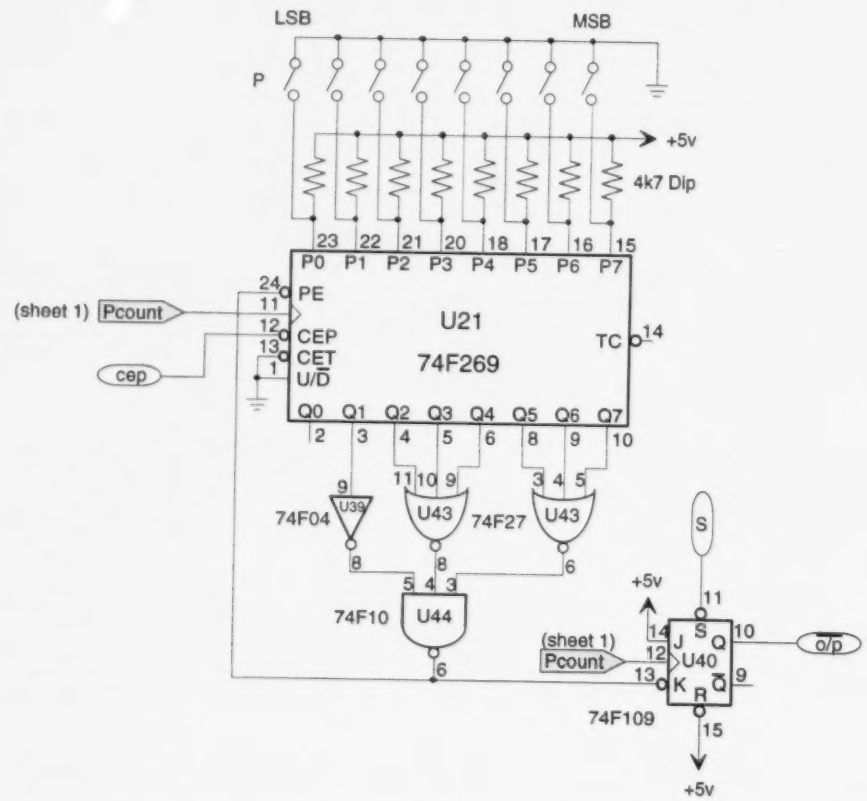
Composite Gapping

Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	Clock Distibution and Composite Gapping	4/11	05/17/94


LSB
MSB
T/8 Select
+5v
4k7 dips
U1
74F269
U5
74F269
P0 P1 P2 P3 P4 P5 P6 P7
PE
U/D
CEP
CET
TC
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
cep
Clk1
(sheet 4)
U2
U7
U3
74F30
U4
74F109
o/p
74F04
Symbol
÷T/8
o/p
cep
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Clock Gapper
Circuit: ÷T/8
Sheet #: 5/11
Date: 05/12/94

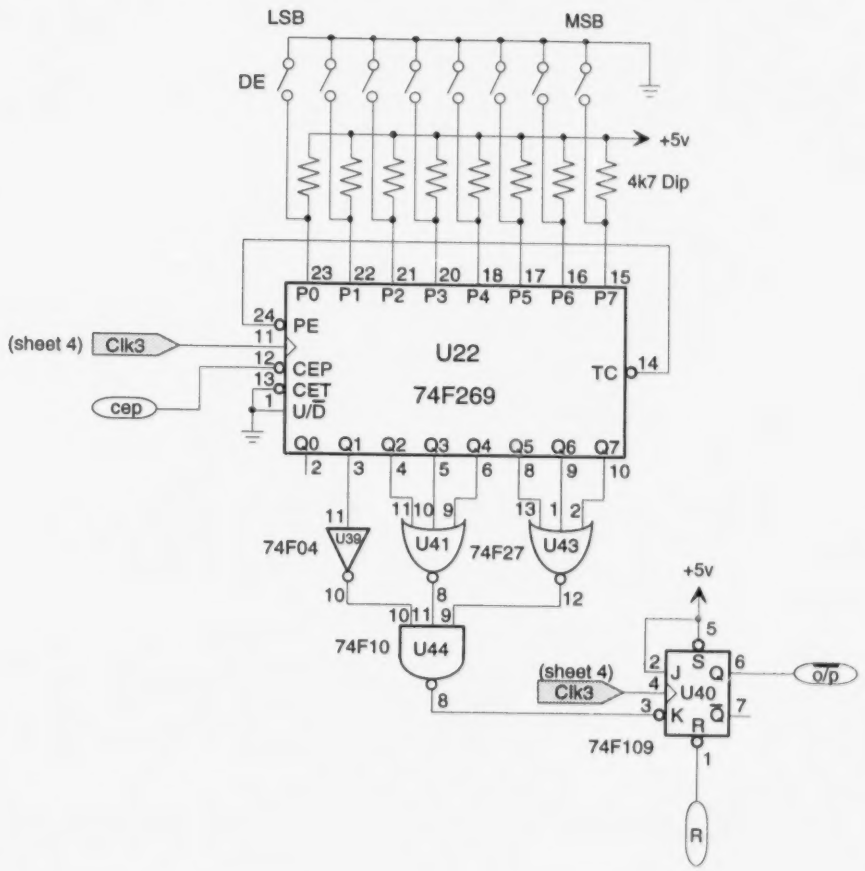

LSB
MSB
DP Select
+5v
4k7 dips
U12
74F269
U16
74F269
P0 P1 P2 P3 P4 P5 P6 P7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
PE
U/D
CEP
CET
TC
cep
Clk2
(sheet 4)
U11
U7
U13
U14
74F30
U15
74F109
o/p
74F04
Symbol
o/p
cep
÷DP
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Clock Gapper
Circuit: ÷DP
Sheet #: 6/11
Date: 05/12/94


LSB
MSB
RP Select
4k7 dips
+5v
U17
74F269
U20
74F269
PE
U/D
CEP
CET
TC
P0 P1 P2 P3 P4 P5 P6 P7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
cep
Clk4
(sheet 4)
U19
74F30
U15
74F109
o/p
Symbol
o/p
cep
+RP
74F04
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Clock Gapper
Circuit: ÷RP
Sheet #: 7/11
Date: 05/12/94



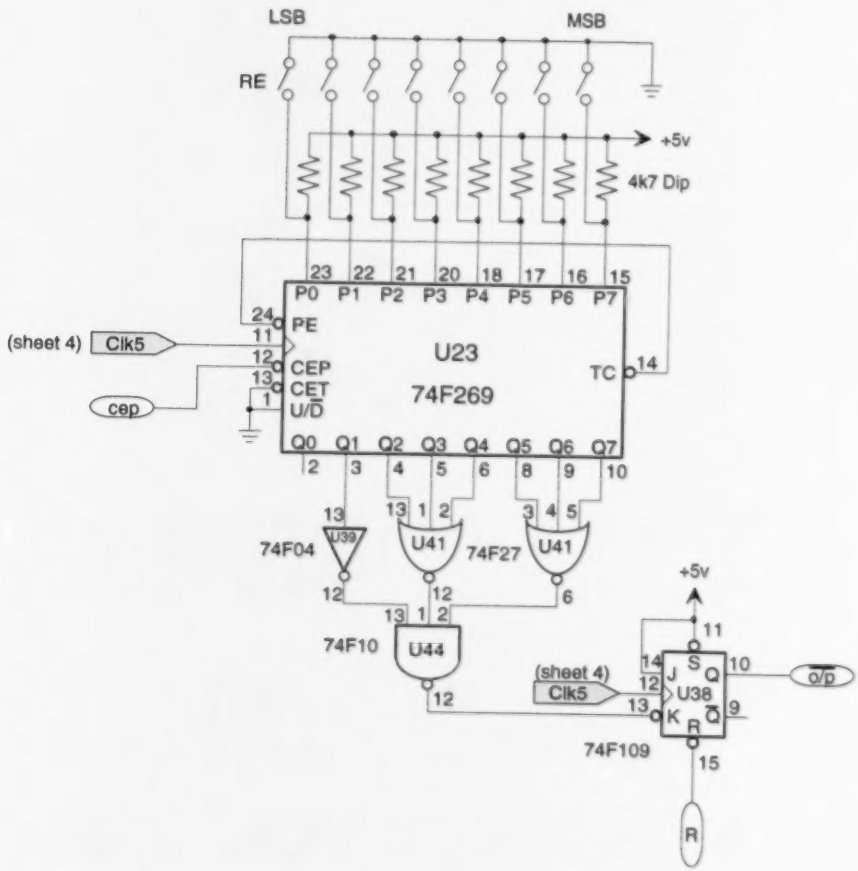
Symbol

Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	÷P	8/11	05/16/94



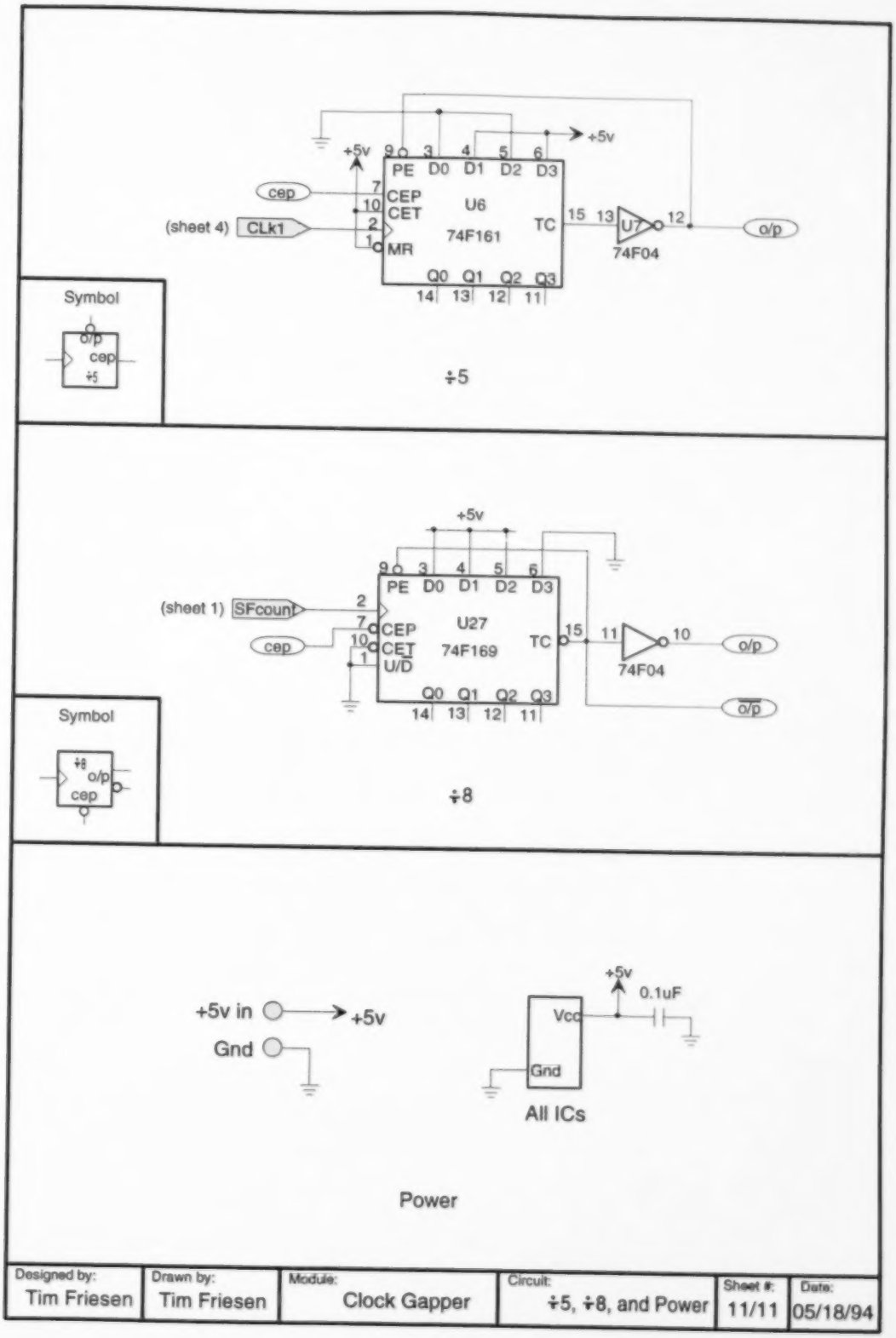
Symbol

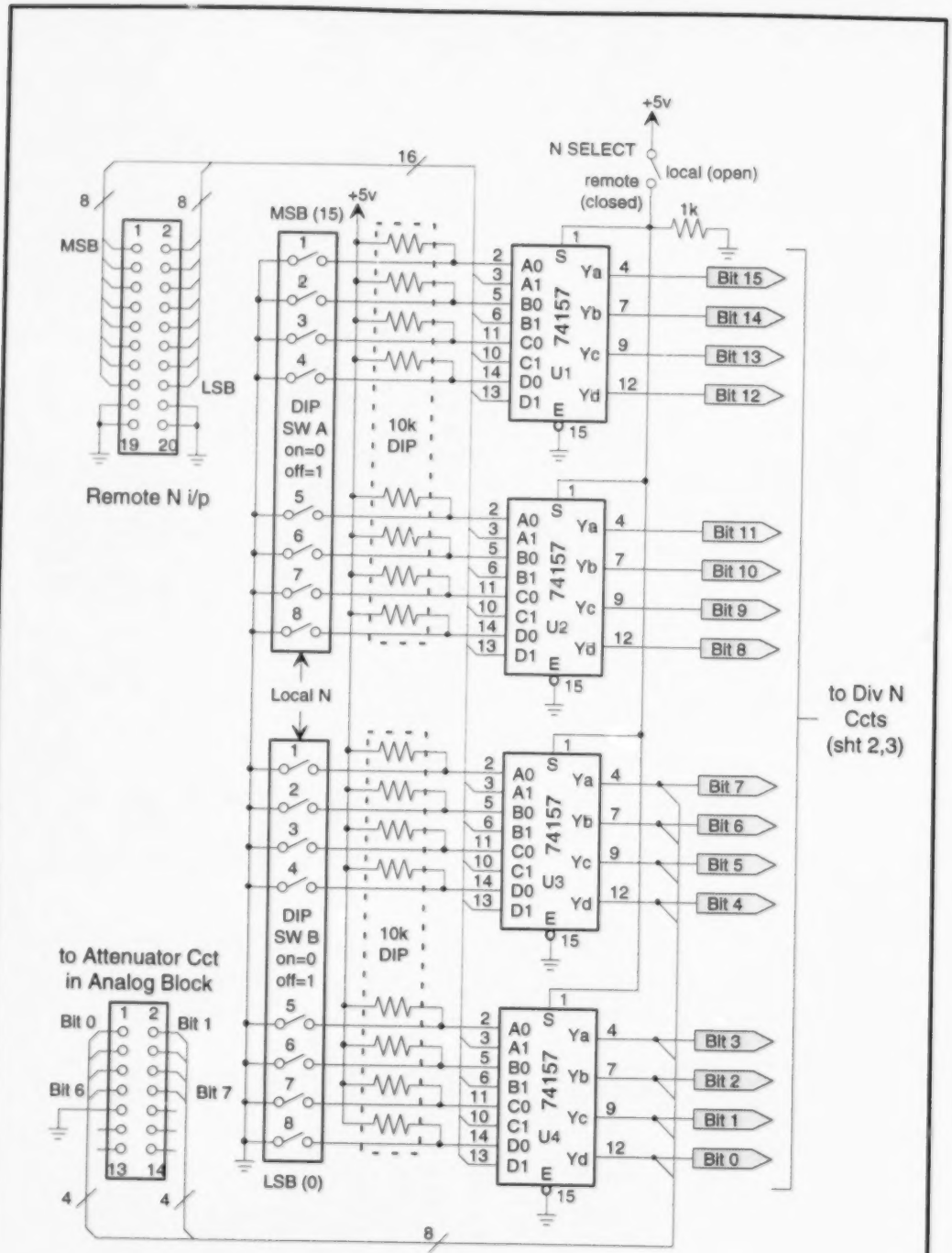
Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	÷DE	9/11	05/16/94



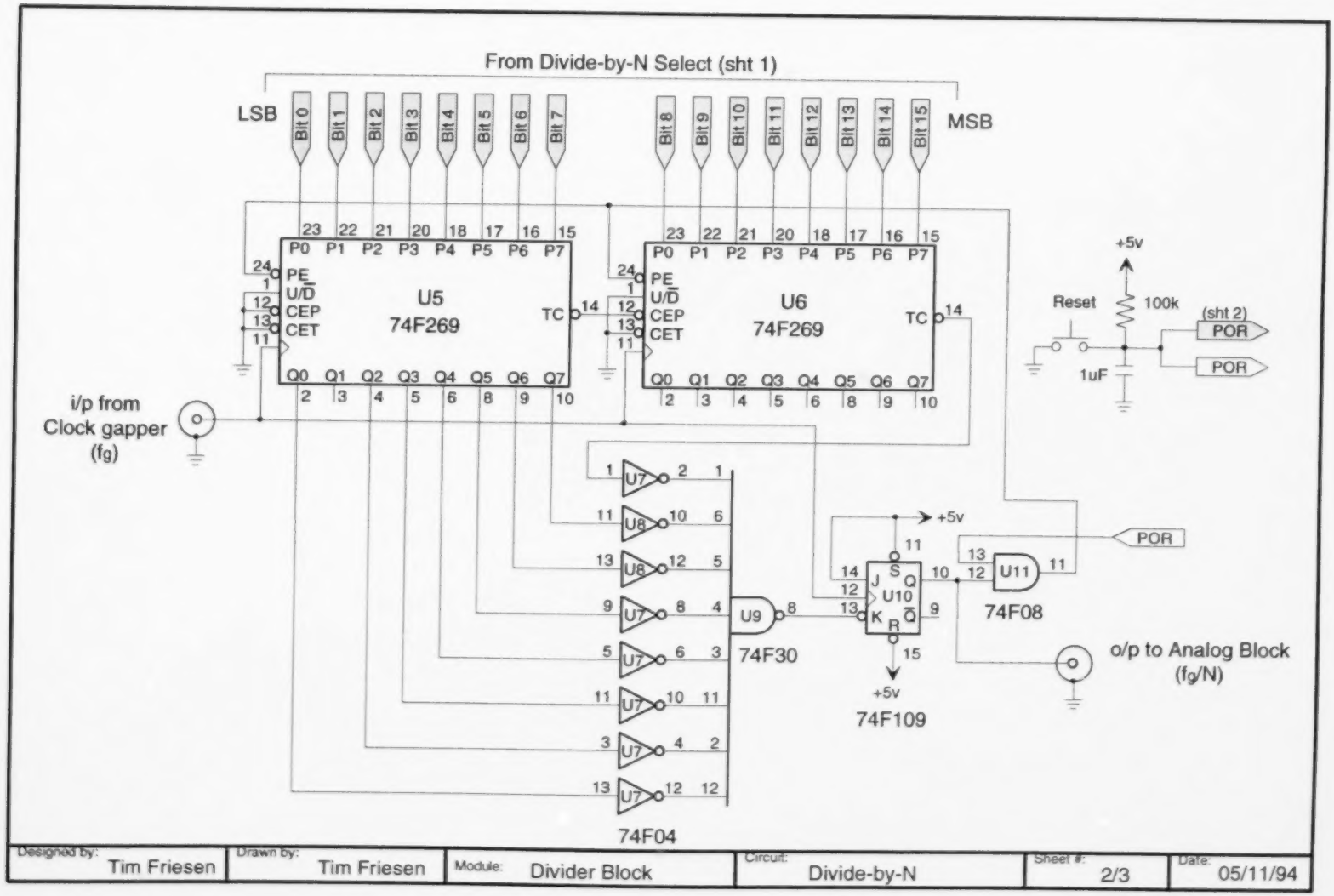
Symbol

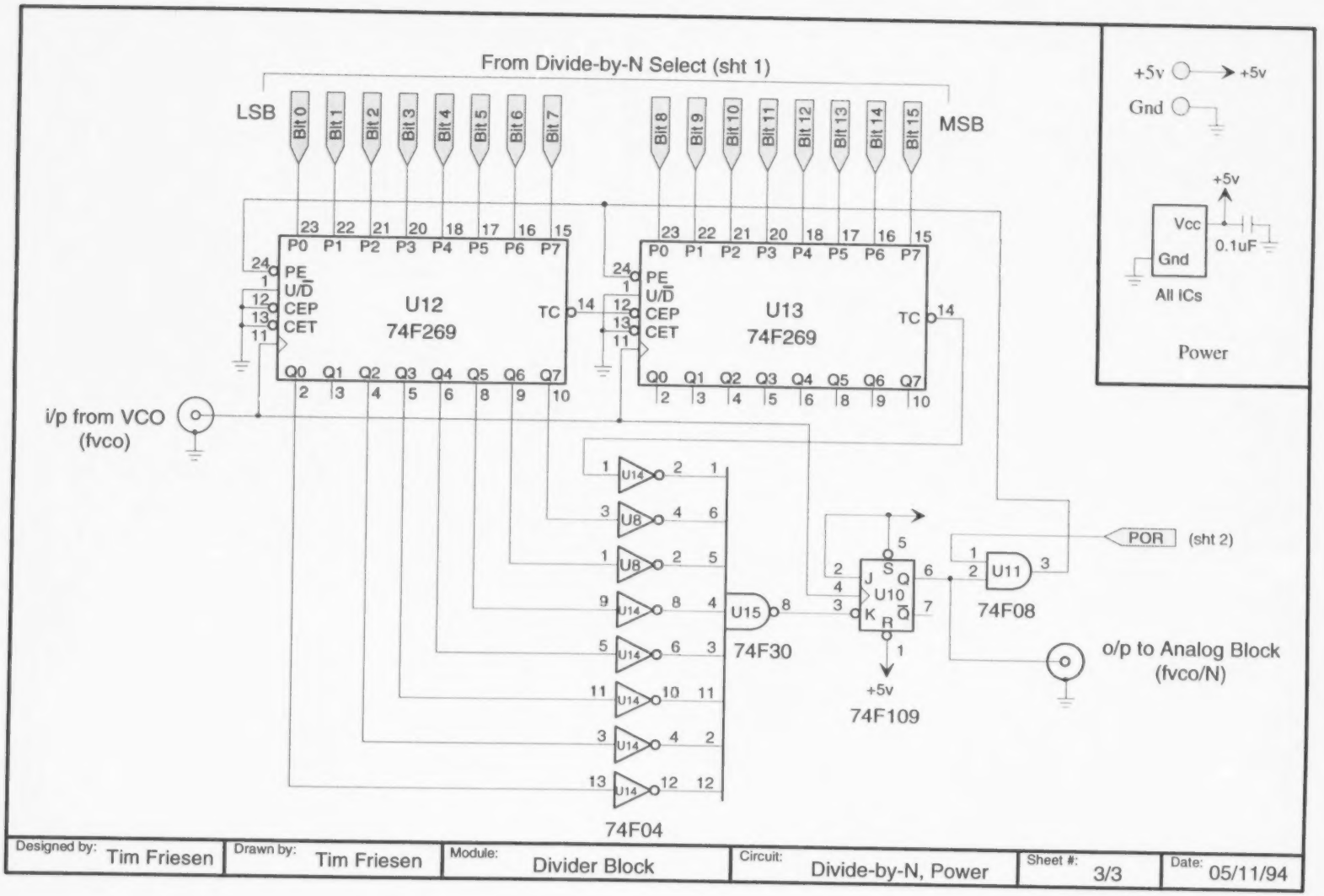
Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Clock Gapper	÷RE	10/11	05/16/94

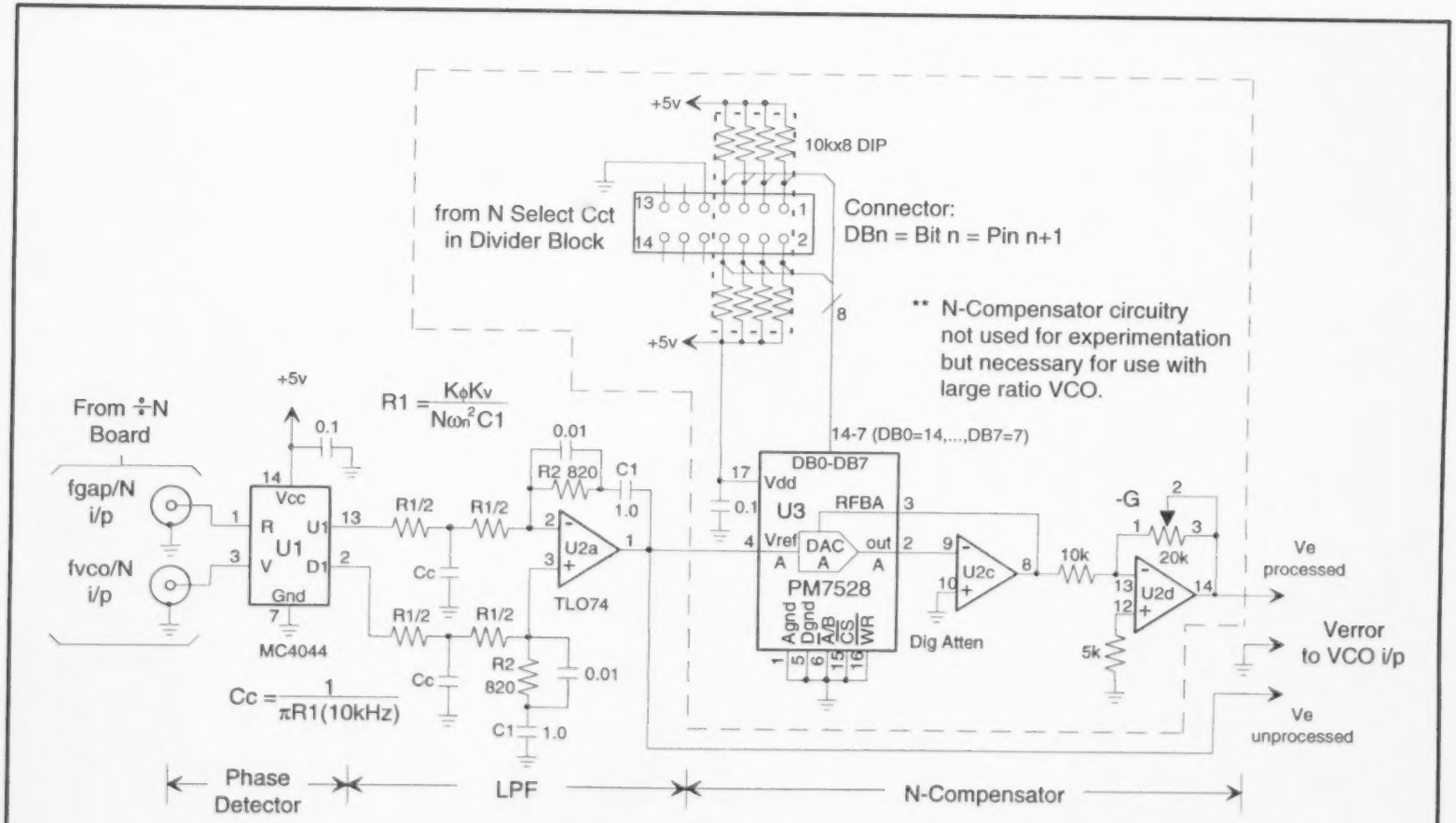

+5v
9 3 4 5 6
PE D0 D1 D2 D3
cep
7
CEP
10
CET
(sheet 4) CLk1
2
1
MR
U6
74F161
TC
15 13
U7
12
74F04
o/p
Q0 Q1 Q2 Q3
14 13 12 11
Symbol
o/p
cep
÷5
÷5
+5v
9 3 4 5 6
PE D0 D1 D2 D3
(sheet 1) SFcount
2
7
CEP
cep
10
CET
1
U/D
U27
74F169
TC
15
11
10
74F04
o/p
o/p
Q0 Q1 Q2 Q3
14 13 12 11
Symbol
÷8
o/p
cep
÷8
+5v in
+5v
Gnd
+5v
0.1uF
Vcc
Gnd
All ICs
Power
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Clock Gapper
Circuit: ÷5, ÷8, and Power
Sheet #: 11/11
Date: 05/18/94



Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Divider Block	N Select	1/3	05/11/94

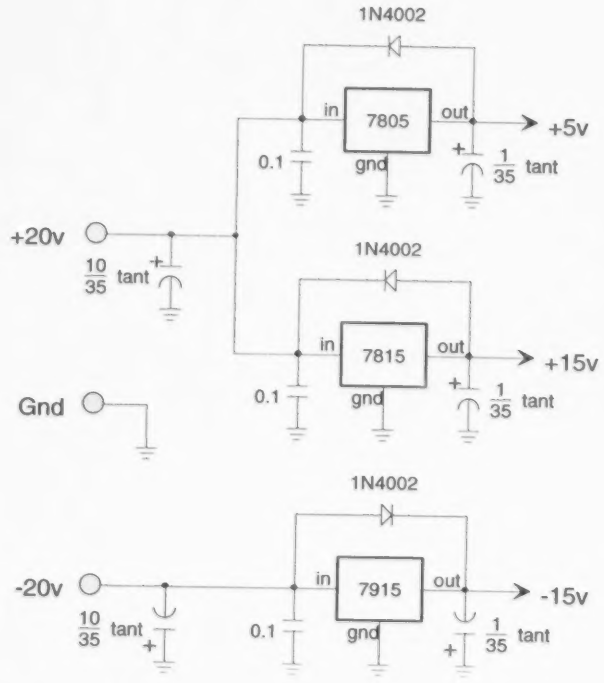

From Divide-by-N Select (sht 1)
LSB
MSB
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15
U5
74F269
U6
74F269
PE
U/D
CEP
CET
TC
P0 P1 P2 P3 P4 P5 P6 P7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
+5v
Reset
100k
1uF
(sht 2)
POR
POR
i/p from
Clock gapper
(fg)
U7
U8
U9
74F30
74F04
U10
74F109
U11
74F08
o/p to Analog Block
(fg/N)
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Divider Block
Circuit: Divide-by-N
Sheet #: 2/3
Date: 05/11/94

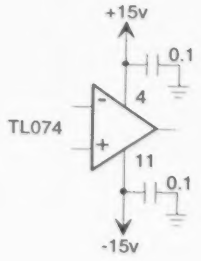

From Divide-by-N Select (sht 1)
LSB
MSB
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15
U12
74F269
U13
74F269
PE
U/D
CEP
CET
TC
P0 P1 P2 P3 P4 P5 P6 P7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
i/p from VCO
(fvco)
U14
U8
U15
74F30
U10
74F109
U11
74F08
POR
(sht 2)
o/p to Analog Block
(fvco/N)
74F04
+5v
Gnd
Vcc
0.1uF
All ICs
Power
Designed by: Tim Friesen
Drawn by: Tim Friesen
Module: Divider Block
Circuit: Divide-by-N, Power
Sheet #: 3/3
Date: 05/11/94



NOTE: All resistances are in ohms, all capacitances are in microfarads.

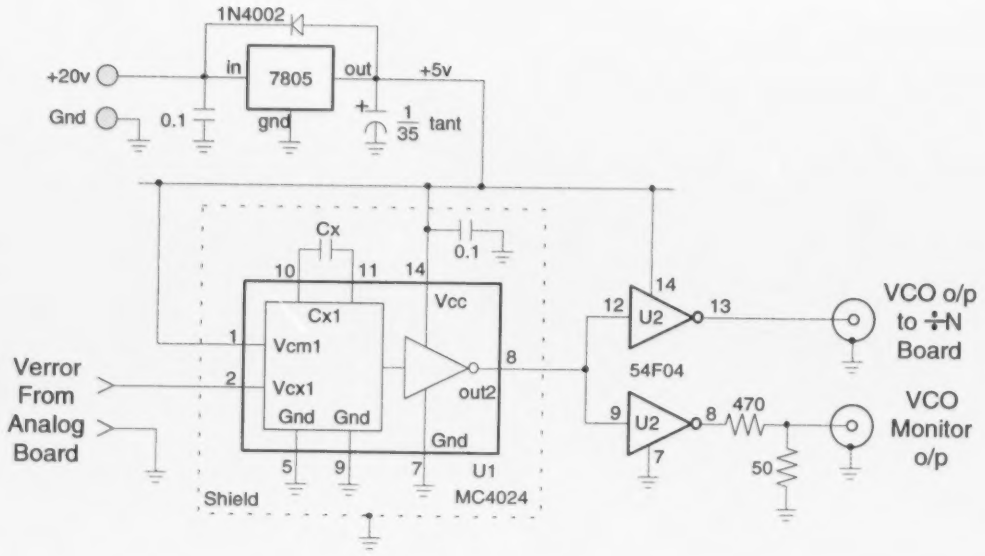
Designed by: Tim Friesen	Drawn by: Tim Friesen	Module: Analog Block	Circuit: PD/LPF/N-Comp	Sheet #: 1/2	Date: 05/05/94





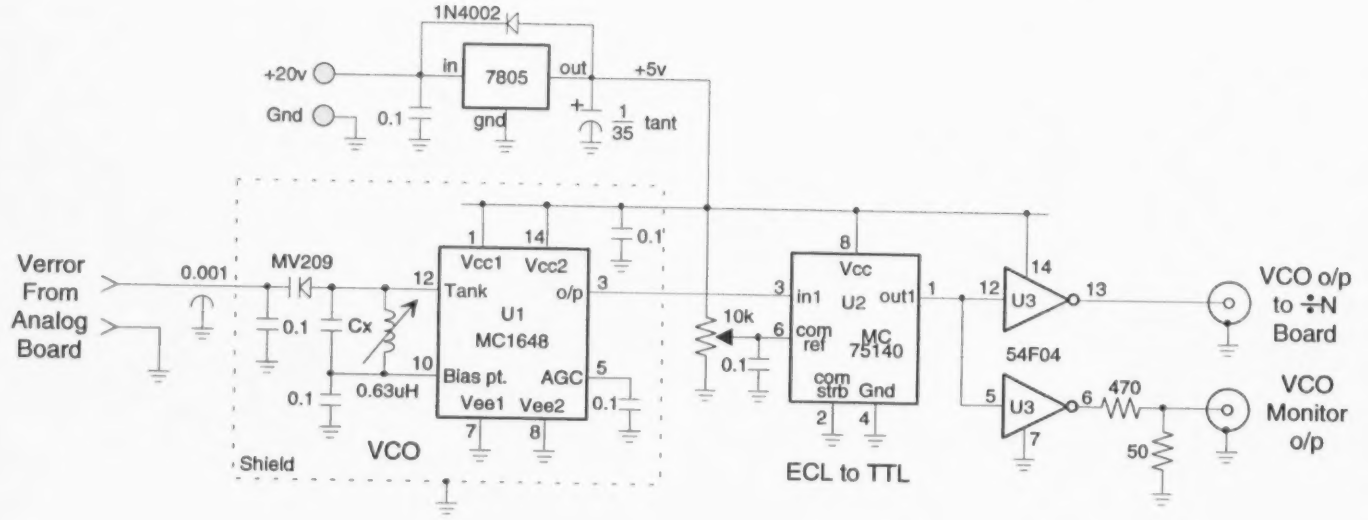
NOTE: All capaciances are in microfarads. Polarized capacitors are expressed as microfarads/rated voltage.

Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Analog Block	Power	2/2	05/05/94



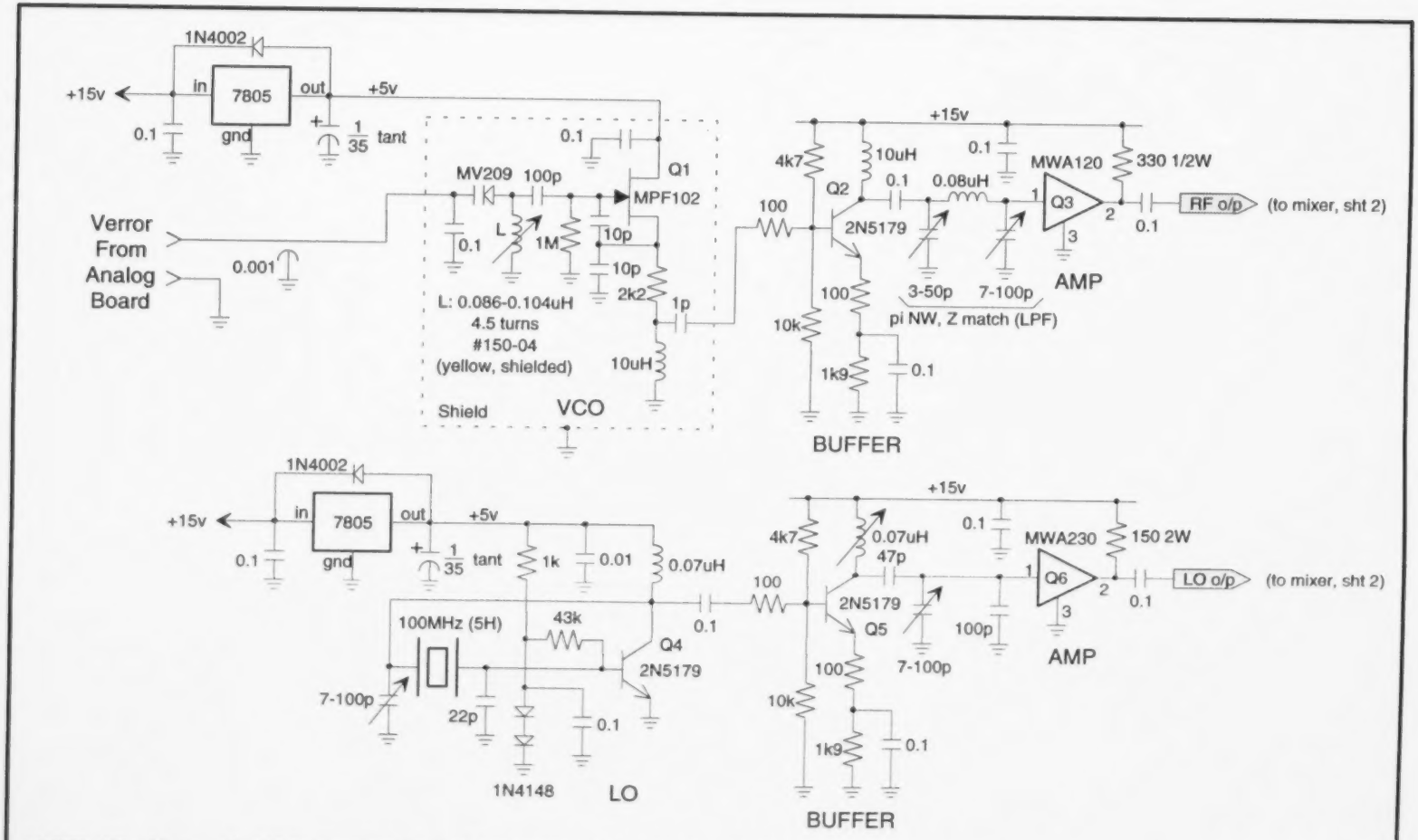
NOTE: All resistances are in ohms, all capacitances are in micofarads.

Designed by: Tim Friesen	Drawn by: Tim Friesen	Module: Small Ratio VCO	Circuit: 4024 VCO (1-5 MHz)	Sheet #: 1/1	Date: 05/11/94



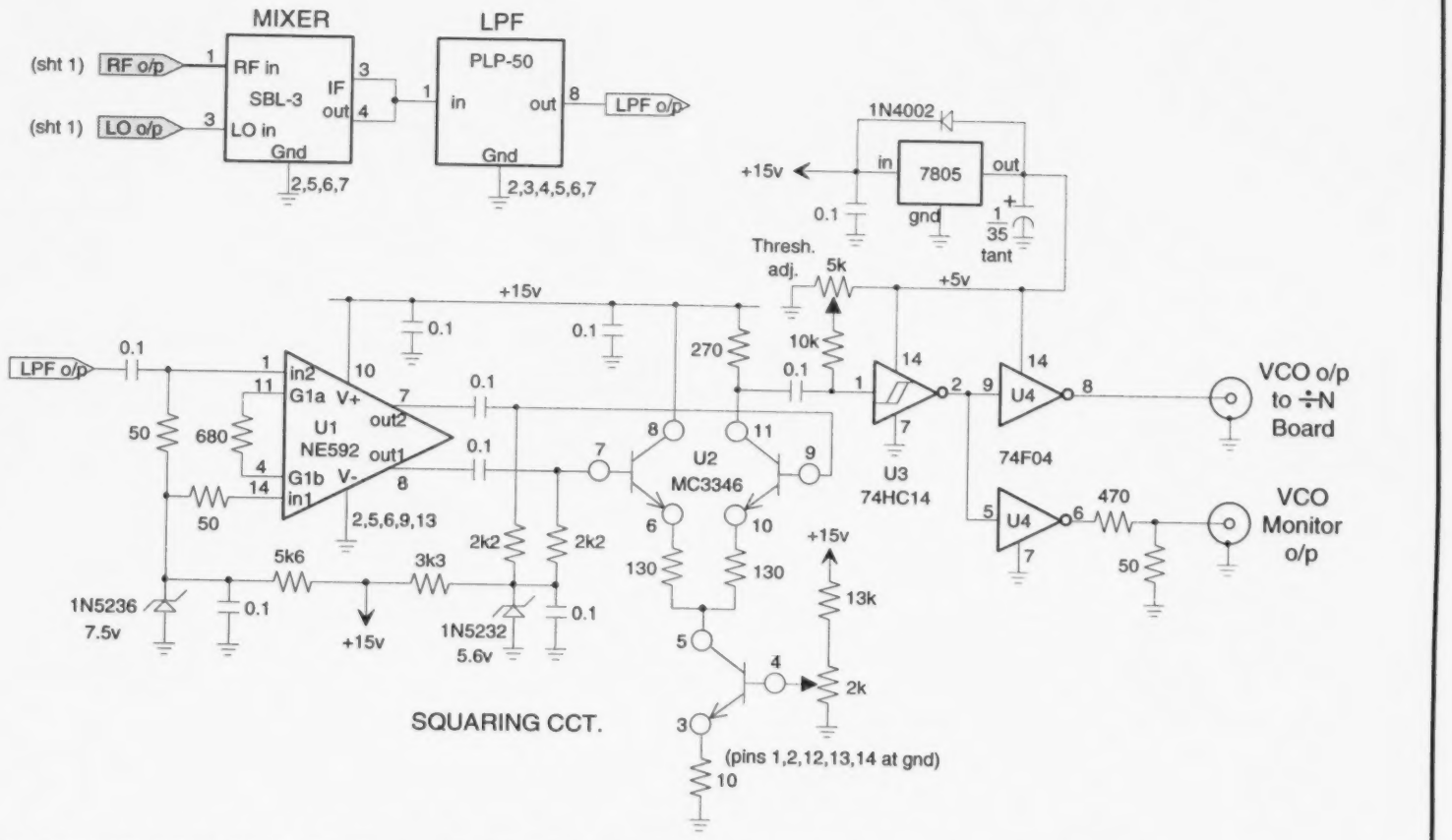
NOTE: All resistances are in ohms, all capacitances are in micofarads.

Designed by: Tim Friesen	Drawn by: Tim Friesen	Module: Small Ratio VCO	Circuit: 1648 VCO (10-40 MHz)	Sheet #: 1/1	Date: 05/11/94



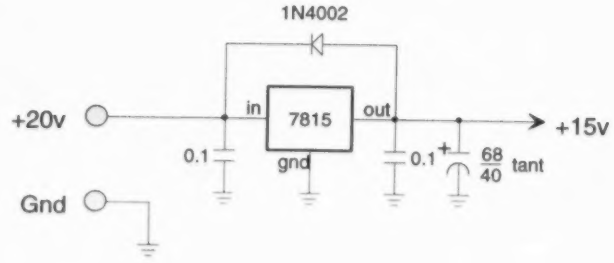
NOTE: The large ratio VCO was constructed but not used for experimentaition.
Unless otherwise stated, all resistances are in ohms, all capacitances are in microfarads (p = picofarads).

Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Large Ratio VCO	VCO/LO	1/3	05/10/94



NOTE: Unless otherwise stated, all resistances are in ohms, all capacitances are in microfarads (p = picofarads).

Designed by: Tim Friesen	Drawn by: Tim Friesen	Module: Large Ratio VCO	Circuit: Mixer/LPF/Squarer	Sheet #: 2/3	Date: 05/10/94



NOTE: All capaciances are in microfarads. Polarized capacitors are expressed as microfarads/rated voltage.

Designed by:	Drawn by:	Module:	Circuit:	Sheet #:	Date:
Tim Friesen	Tim Friesen	Large Ratio VCO	Power Distribution	3/3	05/10/94

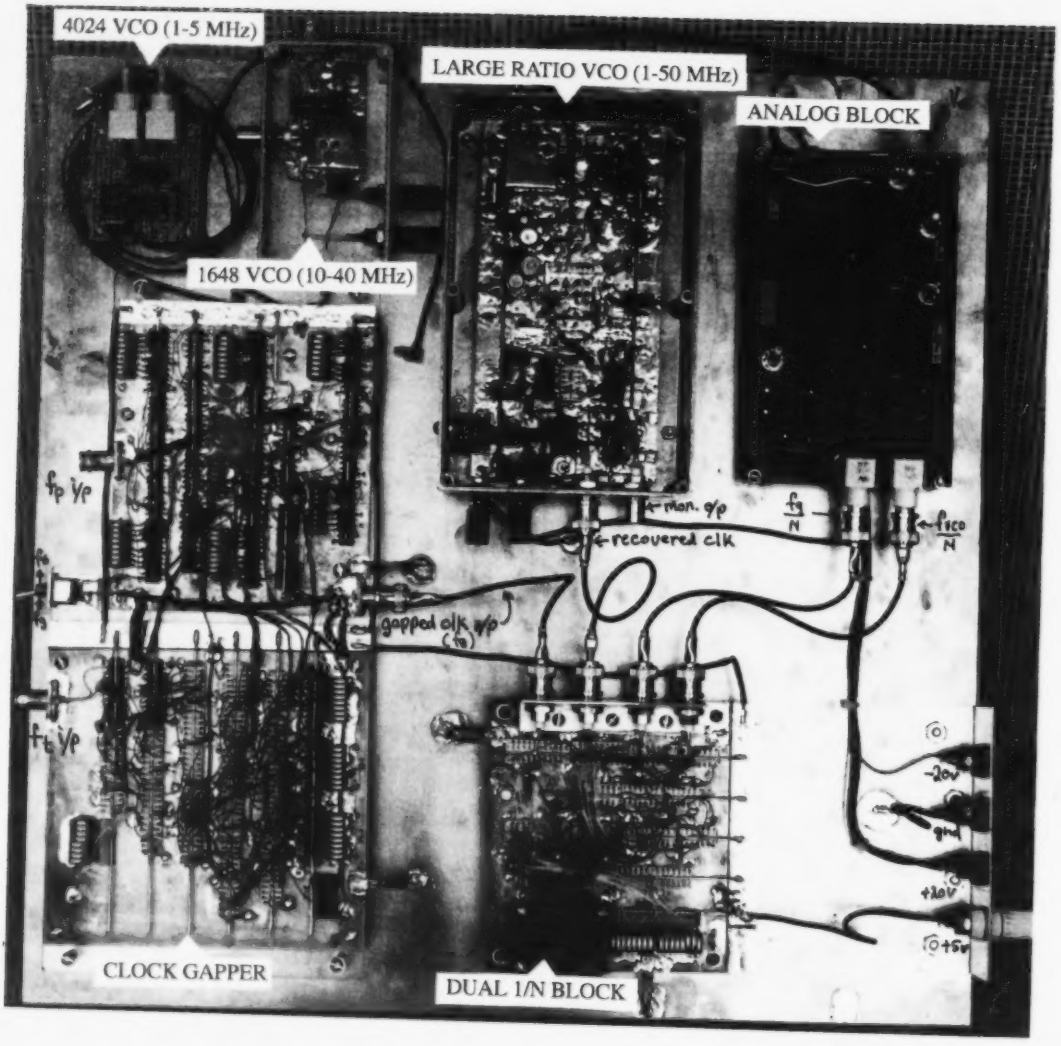


Plate E1 UIPM experimental hardware

APPENDIX F

Derivations for Minimum and Maximum Tributary Rates

This Appendix derives expressions for the theoretical minimum and maximum tributary rates ($f_t$min and $f_t$max) that can be accommodated by the UIPM format design algorithm. Actual values for $f_t$min and $f_t$max are then given for the DS3 payload rate.

Equation 1-1, derived in Appendix A, gives the expression for the stuff ratio (ρ) of a fixed plesiochronous multiplexing format:

$$\rho = \frac{d + so - \frac{N * f_t * T}{f_s}}{so} \tag{1-1}$$

where *so* is the number of stuff opportunities; *d* is the number of time slots in a mapping frame dedicated for carrying tributary data bits; *T* is the total number of bits per frame; *N* is the number of tributaries per frame; f_t is the tributary input rate; and f_s is the synchronous output carrier rate.

Solving for f_t results in:

$$f_t = \frac{(d + so - \rho * so) * f_s}{N * T} \tag{F-1}$$

From this equation the minimum and maximum tributary rates that can be accommodated by a format can be found as follows:

$$f_t \min_{\substack{\rho=1 \\ f_s=f_s \max}} = \frac{d * f_s \max}{N * T}, \quad f_s \max = f_s + \Delta_s f_s \tag{F-2}$$

$$f_t \max_{\substack{\rho=0 \\ f_s=f_s \min}} = \frac{(d + so) * f_s \min}{N * T}, \quad f_s \min = f_s - \Delta_s f_s \tag{F-3}$$

where Δ_s is the tolerance of f_s in ppm. The input offset range offered by the format is then $f_t$max-$f_t$min, while the actual offset range of the tributary signal is $2f_t\Delta_t$, where Δ_t is the

tolerance of f_t in ppm. Obviously the offset range offered by the format must be equal to or greater than (preferably much greater than) the actual offset range of the tributary.

The tolerance of the tributary signal can be included in the expressions for $f_t$min and $f_t$max by replacing $f_t$min with $f_t$min-$\Delta_t f_t$min, and $f_t$max with $f_t$max+$\Delta_t f_t$max. Equations F-2 and F-3 then become:

$$f_t \min_{\substack{\rho=1 \\ f_s=f_s \max}}(\pm\Delta_t) = \frac{d * f_s \max}{N * T * (1 - \Delta_t)} \qquad \text{(F-4)}$$

$$f_t \max_{\substack{\rho=0 \\ f_s=f_s \min}}(\pm\Delta_t) = \frac{(d + so) * f_s \min}{N * T * (1 + \Delta_t)} \qquad \text{(F-5)}$$

In the UIPM generic format N=1 while the number of dedicated data bits (*d*) and the total number of bits per frame (*T*) depend on the format size which can vary. To determine $f_t$min for the UIPM generic format, *d* in Equation F-4 must be the minimum number of data bits per frame (d_{min}) which is 7, while *T* must be the maximum format size (T_{max}) which is 4104. This gives the smallest possible ratio of *d* to *T* and results in the lowest value for $f_t$min.

To determine $f_t$max for the UIPM generic format, there must be a maximum ratio of *d* to *T* which implies there are no fixed stuff bits and a maximum number of data bits. In this case, *T=d+so+oh* where *oh* is the number of dedicated overhead bits per frame which is 40. Therefore, *d+so* in Equation F-5 can be replaced by *T-oh*. A maximum number of data bits requires a maximum format size and thus *T* must be the maximum of 4104 bits.

In the UIPM format design algorithm the carrier payload rate, f_p, is used instead of the full carrier rate (f_s). Incorporating these changes in equations F-4 and F-6 results in the following for the UIPM generic format:

$$\mathbf{f_t\ min_{UIPM}} = \frac{\mathbf{d_{min}} * \mathbf{f_p\ max}}{\mathbf{T_{max}} * (1 - \Delta_t)} \qquad \textbf{(F-6)}$$

$$\mathbf{f_t\ max_{UIPM}} = \frac{\left(1 - \frac{\mathbf{oh}}{\mathbf{T_{max}}}\right) * \mathbf{f_p\ min}}{1 + \Delta_t} \qquad \textbf{(F-7)}$$

For the UIPM generic format using a DS3 carrier payload rate Δ_t=100 ppm and $f_p$min=$f_p$max=44.209694 Mb/s (Δ_p=0). Solving equations F-6 and F-7 with all the numerical substitutions results in **$f_t$min=75.414 kb/s** and **$f_t$max=43.774422 Mb/s** (to the nearest 1 b/s). Software simulation designed to capture all no-format rates (see Chapter 6) verify these values precisely.

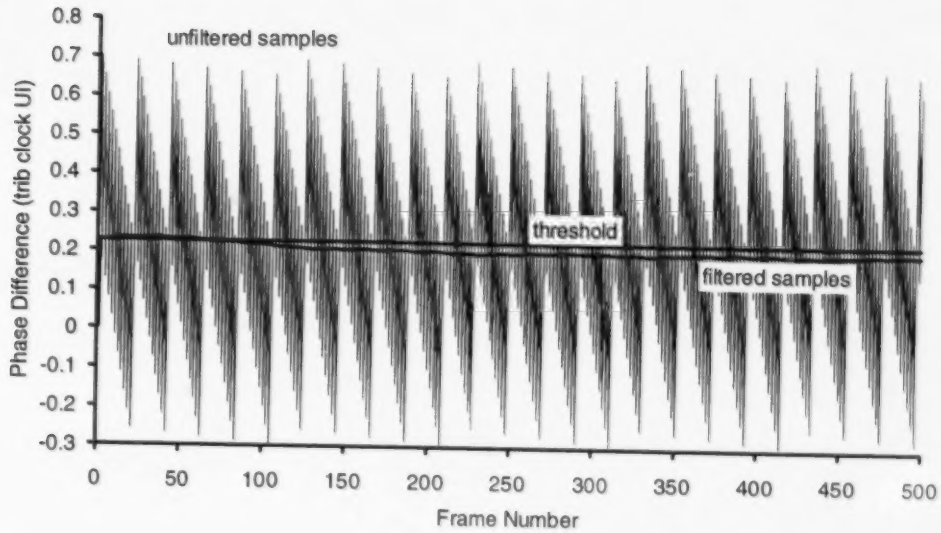
APPENDIX G

Wideband p-p and rms Jitter

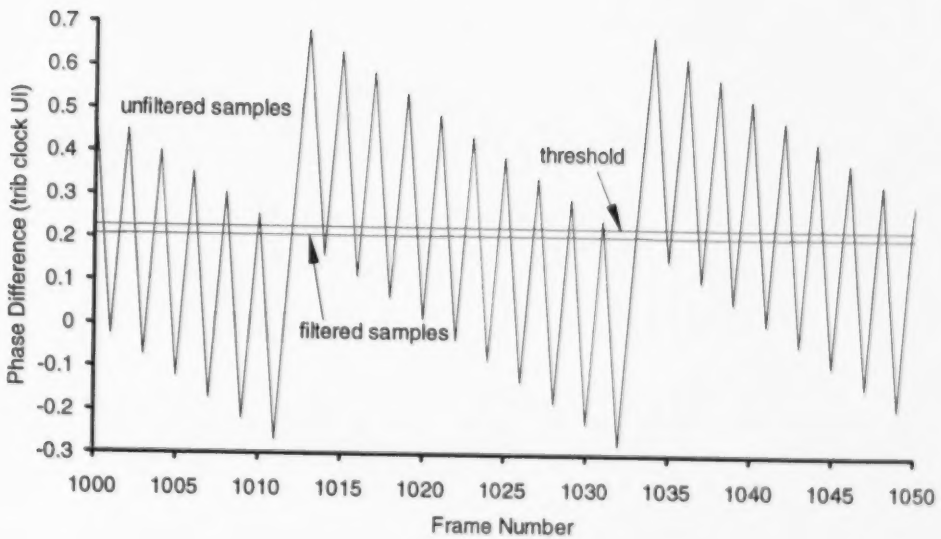
Jitter simulation of positive pulse stuffing justification multiplex schemes for plesiochronous tributaries results in predicted wideband (WB) p-p jitter that is always about 1 UI p-p and WB rms jitter that is always about 0.2886 UI rms. This Appendix attempts to explain mathematically why this is so.

Figures G1 and G2 show the frame-by-frame phase difference (in UI normalized to the tributary, or write, clock) between the read (carrier) and write (tributary) clocks in a positive pulse stuffing synchronizer. The plots in Figure G1 are for a 10 Mb/s signal mapped into a DS3 payload while those in Figure G2 are for the M12 case (four DS1s multiplexed into a DS2 carrier). For both cases, the phase difference was sampled for 2000 frames from the start of simulation. Each set of figures displays a particular range of samples within the entire range of 2000. The first plot of each set shows the phase waveform for the first 500 frames (subsequent frames are similar) while the second plot shows a zoomed in view.

All plots show the unfiltered samples, the filtered samples, and the phase threshold above which a stuff request is initiated. The wideband p-p and rms jitter is derived from statistical operations on the unfiltered samples while the narrowband p-p and rms jitter is derived from statistical operations on the filtered samples.

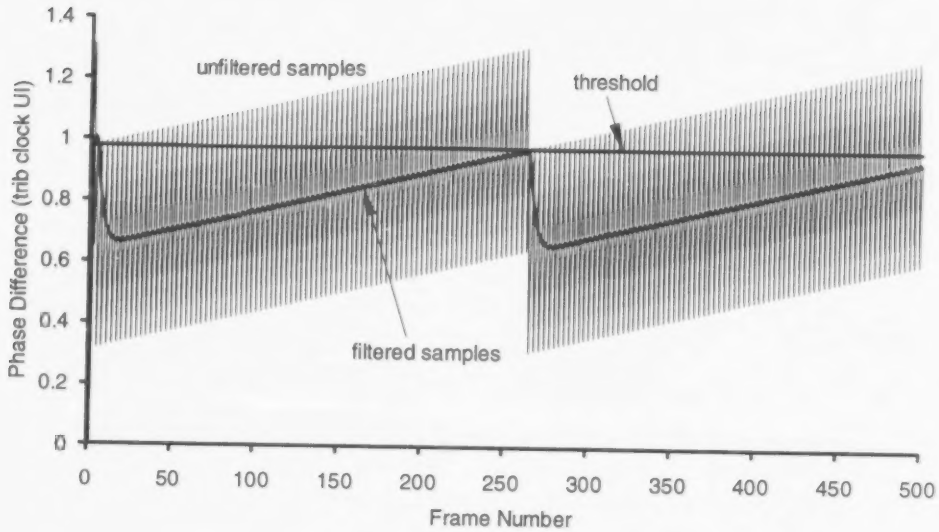


a) Plot of first 500 frames

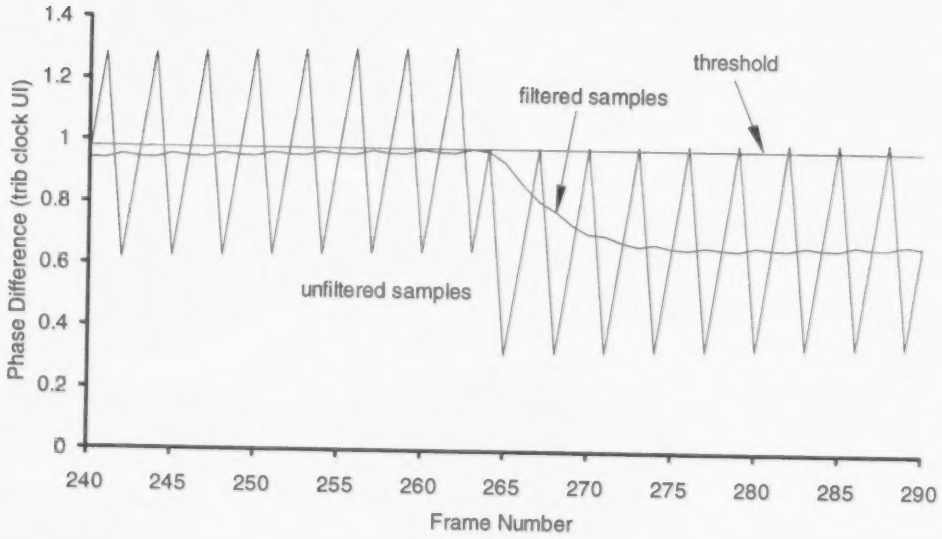


b) Zoom in view

Figure G1 Phase waveform for 10 Mb/s to DS3 payload mapping



a) Plot of first 500 frames



b) Zoom in view

Figure G2 Phase waveform for M12 case

WB p-p Jitter

It is evident from the above figures that the p-p phase difference (i.e. jitter) is indeed about 1 UI. This can be explained mathematically. Following the procedure in [MoKG91], let $\Delta\phi_D$ represent the change in relative phase during one period of the read clock when a time slot contains a data bit, and let $\Delta\phi_O$ represent the change in relative phase during a read clock period when a time slot contains an overhead or stuff bit. The units of $\Delta\phi_D$ and $\Delta\phi_O$ are read clock UI since the carrier time slots are generated by the read clock. Relative phase is defined as the read phase minus the write phase of the two clocks in the synchronizer; this corresponds to t_w-t_r in terms of the write and read clock periods respectively.

During a time slot that contains a data bit the relative phase increases by t_w-t_r in absolute terms. When a time slot contains an overhead or stuff bit the read clock is inhibited for one period and its instantaneous frequency is zero. Thus, the write clock will advance in phase by 1 read clock UI or, conversely, the relative phase will decrease by 1 UI. Therefore, $\Delta\phi_D$ and $\Delta\phi_O$ can be expressed in read clock UI as:

$$\Delta\phi_D = \frac{t_w - t_r}{t_r} = \left(\frac{1}{f_w} - \frac{1}{f_r}\right) \cdot f_r = \frac{f_r}{f_w} - 1 \text{ UI} \qquad \text{(G-1)}$$

$$\Delta\phi_O = \frac{0}{f_w} - 1 = -1 \text{ UI} \qquad \text{(G-2)}$$

where $f_r = \frac{1}{t_r}$ is the read clock frequency and $f_w = \frac{1}{t_w}$ is the write clock frequency.

The p-p phase difference between the read and write clocks is simply $\Delta\phi_D - \Delta\phi_O = \frac{f_r}{f_w}$ in read clock UI because the only difference that exists between any two frames is the status of the stuff bit - it either carries a dummy bit or a data bit. Conversion to write clock UI requires multiplication by $\frac{f_w}{f_r}$ which results in a p-p jitter value of **1 UI**

(write clock). This is essentially the value of predicted WB p-p jitter of the received tributary signal for the UIPM and other related formats.

WB rms Jitter

To explain why the WB rms jitter is always about 0.2886 UI rms, the definition of rms and its implementation in the jitter simulation module must be examined. The jitter simulation module accumulates samples of phase difference values. Only the rms value of the "AC" phase jitter is desired while the "DC" component, or static phase difference, is ignored. For a set of discrete samples this can be computed by determining its standard deviation (σ) which is the square root of the variance (σ^2). The ACrms value of the jitter is then given as:

$$\text{ACrms} = \sigma = \sqrt{\overline{x^2} - \overline{x}^2} \qquad \text{(G-3)}$$

where: x is a random variable; $\overline{x^2}$ is the mean squared value of x; and $\overline{x}^2$ is the square of the mean of x and represents the DC component. For discrete samples of x the calculation for rms jitter becomes:

$$\text{ACrms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)^2} \qquad \text{(G-4)}$$

where N is the number of samples accumulated and x_i is the value of the i^{th} sample. This is the form used in the jitter simulation module of the UIPM format design algorithm.

The envelope of the phase difference waveforms in Figures G1 and G2 represents the low frequency waiting time jitter [Dutt72]. For purpose of this discussion one can assume this is equivalent to a continuous triangular waveform such as a sawtooth (or any other triangular shape) with an amplitude (A) of 1/2 (i.e. 1/2 of the 1 UI p-p) and a period of T as shown in Figure G3.

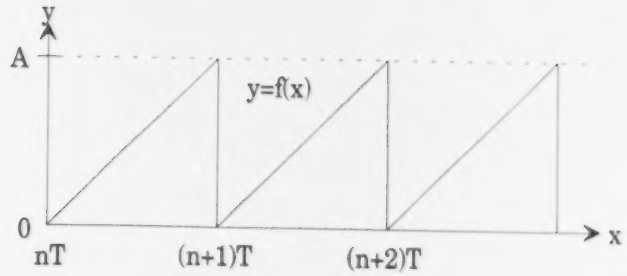


Figure G3 Sawtooth waveform

The equation of a sawtooth waveform over one period is simply $y = \frac{A}{T}x$. The rms value of any periodic function y=f(x) can be found from Equation G-5:

$$y_{rms} = \sqrt{\frac{1}{T}\int_0^T y^2 dx} \qquad \text{(G-5)}$$

Evaluating Equation G-5 for a sawtooth waveform results in:

$$y_{rms} = \sqrt{\frac{1}{T}\int_0^T \frac{A^2}{T^2}x^2 dx} = \sqrt{\frac{A^2}{T^3}\cdot\frac{x^3}{3}\Big|_0^T} = \sqrt{\frac{A^2}{3}} = 0.577350A \qquad \text{(G-6)}$$

For a sawtooth waveform with an amplitude of A=1/2, $\mathbf{y_{rms}=0.288675}$ which is essentially the value of the predicted WB rms jitter of the recovered tributary signal for UIPM and other related formats.

One could also assume the phase difference waveforms are equivalent to a triangular waveform with an amplitude of 1 UI p-p. Then the continuous form of Equation G-4 would result in exactly the same value.

APPENDIX H

Glossary

adapting the process of converting an isochronous digital signal into a higher rate signal of some particular format
e.g. UIPM is the process that adapts arbitrary-rate isochronous digital signals into the payload portion of a standard carrier.

arbitrary-rate signal a signal whose rate is fixed at some arbitrary value but may deviate from that rate within a specified tolerance

asynchronous two or more signals are asynchronous if there is no phase relationship between their corresponding significant instants

asynchronous communication the transfer of information using digital signals that do not have a constant bit rate and are thus not isochronous signals

carrier a standard telecommunications carrier such as a DS3 or STS-3

carrier format the format of a standard telecommunications carrier, such as a DS1 or DS3, that describes the position and function of all bits in each frame of the format

carrier payload the portion of a carrier's format dedicated to carriage of payload data

carrier payload rate the aggregate rate of the carrier's payload data
e.g. This is the number of carrier payload bits times the carrier's frame rate. In a UIPM system, the rate of a UIPM signal is equal to the carrier payload rate.

carrier signal rate the signal rate of a standard carrier (e.g. 44.736 Mb/s for DS3)

frequency offset range the range in Hz that a signal may deviate from its specified nominal value

frequency tolerance the amount in parts-per-million (ppm) a signal's rate is allowed to deviate from its specified nominal value

isochronous a signal is isochronous if the time interval separating any two significant instants is theoretically equal to the unit interval or to a multiple of the unit interval (in practice, variation in the time intervals are constrained within specified limits)

jitter short term variations of the significant instants of a digital signal from their ideal positions in time

mapping the process of placing payload data into the appropriate time slots of a format e.g. An arbitrary-rate tributary is the payload for a UIPM format, while the resulting UIPM signal is the payload for a standard carrier.

muldem a multiplexer/demultiplexer system

overhead extra bits of a format, in addition to payload data bits, that provide framing and other information to terminating equipment

plesiochronous the relationship that exists between two or more signals when their rates are nominally the same but may vary within specified limits

plesiochronous multiplexing the multiplexing of plesiochronous tributaries into a higher rate carrier by first converting each tributary into a slightly higher rate synchronous signal through the use of pulse stuffing justification

public network the part of the telecommunications network used to service the public sector

pulse stuffing justification the action of inhibiting a read from a buffer (or elastic store) when the read clock threatens to overtake the write clock, and replacing the data bit that would have normally been read with a *dummy* or *stuff* bit during the time slot allocated as the variable stuff opportunity

standard carrier a signal, such as a DS3 or STS-3, whose rate and format is standardized in the telecommunications industry, and whose purpose is to carry or transport voice or data

synchronous two signals are synchronous if their corresponding significant instants have a desired phase relationship

tributary one or more signals that are multiplexed together along with necessary overhead bits to form another higher rate signal of some specified format e.g. 7 DS1 tributaries are multiplexed together to form a DS3; a single tributary is multiplexed with UIPM overhead bits to form a UIPM-formatted signal.

UIPM demultiplexer the portion of a UIPM system that recovers the original tributary data

UIPM format a description of the position and function of all the bits used in a frame of a particular UIPM format design that is optimal for the tributary data it is adapting into the payload of a standard carrier

UIPM format design algorithm an algorithm that determines the optimum size of a variable length format used to adapt an arbitrary-rate isochronous digital signal into to payload portion of a standard carrier

UIPM generic format the generic structure of the UIPM variable length format

UIPM multiplexer the portion of a UIPM system that multiplexes tributary data with UIPM overhead bits to create payload for a standard carrier

UIPM payload the original customer tributary data

UIPM signal rate equal to the standard carrier payload rate of the carrier used

UIPM variable length format a term describing the UIPM generic format whose length or size is determined by the UIPM format design algorithm

variable stuff opportunity a dedicated time slot in a plesiochronous multiplexing format that can carry either a payload data bit or a dummy bit (for use with pulse stuff justification)

waiting time jitter the random waiting period that exists between a stuff request and the next available variable stuff opportunity in plesiochronous multiplexing systems